LS-DYNA's Incompressible Flow Solver User's Manual

MARK A. CHRISTON, Ph.D.,
and
GRANT O. COOK JR., Ph.D.

March, 2001

Version 960
Development version

Copyright ©2001
LIVERMORE SOFTWARE
TECHNOLOGY CORPORATION
All Rights Reserved

Mailing address:
Livermore Software Technology Corporation
2876 Waverley Way
Livermore, California 94550-1740

Support Address:
Livermore Software Technology Corporation
7374 Las Positas Road
Livermore, California 94550

FAX: 925-449-2507
TEL: 925-449-2500
EMAIL: sales@lstc.com

Copyright ©2001 by Livermore Software Technology Corporation
All Rights Reserved

Contents

1 Introduction **1**

1.1 Guide to the User's Manual 2

2 Governing Equations **3**

2.1 Conservation Equations 3

2.1.1 Momentum Conservation 3

2.1.2 Mass Conservation 4

2.1.3 Energy Conservation 5

2.1.4 Boundary and Initial Conditions 5

2.2 Conservation Equations – Vector Notation 7

2.2.1 Momentum Conservation 7

2.2.2 Mass Conservation 8

2.2.3 Energy Conservation 9

2.2.4 Boundary and Initial Conditions 9

3 Explicit Time Integration **13**

3.1 Modified Finite Element Formulation 16

3.1.1 Reduced-Integration Operators 17

3.2 The Domain-Decomposition Message-Passing Paradigm 21

3.2.1 Domain-Decomposition 22

3.2.2 Parallel Assembly via Message Passing 22
3.2.3 The Parallel Explicit Algorithm 24
3.2.4 Communication Costs 25

4 The Projection Method **27**

4.1 Semi-Implicit Projection Method 29
4.2 Fully-Implicit Projection Method 32
4.2.1 Start-Up 32
4.2.2 First Time-Step 33
4.2.3 General Time-Step 33

5 Boundary Conditions and Source Terms **35**

5.1 Node and Segment Sets 35
5.2 Nodal Boundary Conditions 35
5.3 Traction Boundary Conditions 37
5.3.1 Pressure Boundary Conditions 37
5.4 Outflow Boundary Conditions 38
5.5 Body Forces 38
5.6 Flux Boundary Conditions 39
5.7 Pressure Levels 40

6 Pressure Solution Methods **41**

6.1 The Saddle Point Problems 41
6.2 $Q1Q0$ Stabilization 42
6.3 The Projection CG Method 44
6.4 PPE Solver Performance 47

7 Turbulence Models **57**

7.1 Averaging, Filtering and Scale Separation 58

7.1.1 Spatial Filtering 59

7.2 The Time-Averaged Equations 60

7.3 The Spatially Filtered Equations 61

7.3.1 Baseline Smagorinsky Subgrid-Scale Model 62

7.4 Subgrid Kinetic Energy Models 62

7.4.1 Local Dynamic k^{sgs} model 63

8 Flow Statistics **65**

8.1 Derived Flow Statistics 65

8.1.1 Reynolds Averaged Statistics 66

8.1.2 Mean and Fluctuating Quantities 67

8.1.3 Higher-Order Statistics 68

8.1.4 Anisotropic Stress Tensor 69

8.2 LS-DYNA Statistics Levels 70

8.2.1 Level-1: Mean Statistics 71

8.2.2 Second Moment Statistics 71

8.2.3 Higher-Order Statistics 72

8.3 LS-POST Statistics Levels 73

9 Keyword Input **75**

9.1 Incompressible Flow Keywords 75

9.1.1 ***BOUNDARY_OUTFLOW_CFD_***OPTION* 76

9.1.2 ***BOUNDARY_PRESCRIBED_CFD_***OPTION* 77

9.1.3 ***BOUNDARY_PRESSURE_CFD_SET** 79

9.1.4 ***CONTROL_CFD_AUTO** 81

9.1.5 ***CONTROL_CFD_GENERAL** 83

9.1.6 ***CONTROL_CFD_MOMENTUM** 85

9.1.7 ***CONTROL_CFD_PRESSURE** 88

9.1.8 ***CONTROL_CFD_TRANSPORT** 90

9.1.9 ***CONTROL_CFD_TURBULENCE** 94

9.1.10 ***INITIAL_CFD** 95

9.1.11 ***MAT_CFD**_*OPTION* 97

9.2 Shared Keywords 99

10 Example Problems **101**

10.1 Poiseuille Flow 101

10.2 Natural Convection in a Square Cavity 111

10.3 The Momentum-Driven Jet 121

10.4 The Sheath-Flow Chamber 131

A Sense Switch Controls **139**

Preface

The incompressible flow solver in LS-DYNA originated from research conducted by the author at Lawrence Livermore National Laboratory and Sandia National Laboratories during the 1990s. This research was done in collaboration with Phil Gresho and Stevens Chan at Lawrence Livermore National Laboratory. The author is indebted to Phil Gresho for his support and his willingness to share his insights into the behavior of time-dependent incompressible flows. Many of the algorithmic aspects of the explicit flow solution algorithm derive directly from John Hallquist's pioneering work with low-order finite elements, and reduced integration in DYNA3D. I would like to acknowledge Nielen Stander, Ian Do, Jason Wang, Phil Gresho and Steve Sutton at Lawrence Livermore National Laboratory, and Tom Voth at Sandia National Laboratories for their helpful suggestions during the preparation of this manual. In addition, I would like to thank Suresh Menon at Georgia Tech for his assistance with the k^{sgs} LES models, and Sara Lucas for her help with the turbulence statistics. Finally, I would like to thank John Hallquist for providing the opportunity to contribute to LS-DYNA's growing set of simulation capabilities.

Mark A. Christon
Albuquerque, NM
March, 2001

Chapter 1

Introduction

The incompressible flow capability in LS-DYNA has been developed specifically to attack the class of transient, incompressible, viscous, fluid dynamics problems that are predominant in the world that surrounds us. The goal for for the flow solver has been to achieve high performance across a spectrum of supercomputer architectures without sacrificing any of the aspects of the finite element method that make it so flexible and permit application to a broad class of fluid dynamics problems.

The incompressible flow solver plays two primary roles in LS-DYNA. The first is to provide a stand-alone incompressible flow solver that complements the existing solid, structural, boundary-element, and compressible flow capabilities that comprise LS-DYNA's multi-physics capabilities. The second is to provide the flow-solver for fluid-solid/structure interaction problems where the flow is in the low-Mach/incompressible regime.

LS-DYNA's incompressible flow solver is based, in part, upon the work of Gresho, et al. [1, 2, 3, 4], and makes use of advanced solution algorithms for both implicit and explicit time integration. The explicit solution algorithm [1, 2] sacrifices some phase accuracy, but decouples the momentum equations and minimizes the memory requirements. While both the diffusive and Courant-Freidrichs-Levy (CFL) stability limits must be respected in the explicit algorithm, balancing tensor diffusivity meliorates the restrictive diffusive stability limit and raises the order of accuracy of the time integration scheme. The explicit algorithm, in combination with single point integration and hourglass stabilization, has proven to be both simple and efficient in a computational sense.

In the second-order projection algorithm [3, 4], a consistent-mass predictor in conjunction with a lumped mass corrector legitimately decouples the velocity and pressure fields thereby reducing both memory and CPU requirements relative to more traditional fully coupled solution strategies for the Navier-Stokes equations. The consistent mass predictor retains fourth-order advective phase accuracy, while the lumped mass corrector (a projection to a divergence-free subspace) maintains a divergence free velocity field. Both the predictor and the corrector steps are amenable to solution via direct or preconditioned iterative techniques making it possible to tune the algorithm to the computing platform, i.e., parallel, vector or super-scalar. The second-order projection algorithm can accurately track shed vortices, and is amenable to the incorporation of either simple or complex (multi-equation) turbulence sub-models appropriate for a broad spectrum of applications.

1.1 Guide to the User's Manual

The purpose for this document is to provide sufficient information for an experienced analyst to use LS-DYNA's flow solver in an effective way. The assumption is that the user is somewhat familiar with modern computing practices, common CFD practices, and to a certain degree, the current CFD literature. This manual provides sufficient references to the literature to permit the interested reader to pursue the technical details of LS-DYNA.

In Chapters 2–6, an overview of the theoretical background for LS-DYNA is presented. Chapters 7 and 8 provide an overview of the turbulence modeling and the computation of derived flow statistics. Chapter 9 presents information on LS-DYNA's keyword input. Several sample calculations are presented in Chapter 10 to provide guidance for new users.

Chapter 2

Governing Equations

This chapter presents the basic forms of the partial differential equations that LS-DYNA's flow solver treats, and Chapters 3 – 4 provide a general description of the methodologies employed in their solution. The interested reader may pursue the references included in these chapter for details on the incompressible flow solution algorithms used in LS-DYNA and their implementation.

2.1 Conservation Equations

In the ensuing discussion, indicial notation is used with repeated subscripts indicating summation unless otherwise noted. The conservation equations are also presented in vector notation in §2.2.

2.1.1 Momentum Conservation

To begin, the conservation of linear momentum is

$$\rho \frac{\partial u_i}{\partial t} + \rho u_j \frac{\partial u_i}{\partial x_j} = \frac{\partial \sigma_{ij}}{\partial x_j} + \rho f_i, \tag{2.1}$$

where u_i is the velocity, σ_{ij} is the stress tensor, ρ is the mass density, and f_i is the body force. The body force contribution ρf_i typically accounts for buoyancy forces with f_i representing the acceleration due to gravity.

The stress may be written in terms of the fluid pressure and the deviatoric stress tensor as

$$\sigma_{ij} = -p\delta_{ij} + \tau_{ij}, \tag{2.2}$$

where p is the pressure, δ_{ij} is the Kronecker delta, and τ_{ij} is the deviatoric stress tensor.

A constitutive equation relates the deviatoric stress and the strain rate, e.g.,

$$\tau_{ij} = 2\mu_{ij} S_{ij}. \tag{2.3}$$

Here, the dynamic viscosity, μ_{ij}, is represented as a second-rank tensor with no summation on i, j in Eq. (2.3). Frequently, the fluid viscosity is only available as an isotropic viscosity, in which case, the constitutive relation becomes

$$\tau_{ij} = 2\mu S_{ij}. \tag{2.4}$$

The strain-rate tensor is written in terms of the velocity gradients as

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right). \tag{2.5}$$

2.1.2 Mass Conservation

The mass conservation principle in divergence form is

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u_j)}{\partial x_j} = 0. \tag{2.6}$$

In the incompressible limit, the velocity field is solenoidal,

$$\frac{\partial u_i}{\partial x_i} = 0, \tag{2.7}$$

which implies a mass density transport equation,

$$\frac{\partial \rho}{\partial t} + u_j \frac{\partial \rho}{\partial x_j} = 0. \tag{2.8}$$

For constant density, Eq. (2.8) is neglected with Eq. (2.7) remaining as a constraint on the velocity field.

LS-DYNA provides the capability to transport up to 10 species with $Z_1, Z_2, ..., Z_{10}$ representing the 10 species mass concentrations. In order to simplify the presentation, a single mass fraction is presented representing a binary mixture. Mass conservation applied to one species yields for Z_1

$$\rho \frac{\partial Z_1}{\partial t} + \rho u_i \frac{\partial Z_1}{\partial x_i} = -\frac{\partial J_{1_i}}{\partial x_i} + \dot{m}_1, \tag{2.9}$$

where J_{1_i} is the diffusional mass flux rate, and $\dot{m}_1$ is a volumetric mass source. The mass source may include the injection of mass concentration from a boundary or the source/sink terms from chemical reactions.

The diffusional mass flux rate is based on Fick's law of diffusion,

$$J_{1_i} = -\rho D_{1_{ij}} \frac{\partial Z_1}{\partial x_j}, \tag{2.10}$$

where $D_{1_{ij}}$ is a tensorial mass diffusivity. Typically mass diffusivities are only available as scalars so that

$$J_{1_i} = -\rho D_1 \frac{\partial Z_1}{\partial x_i}. \tag{2.11}$$

In the most general form, the species concentration transport equations are

$$\rho\frac{\partial Z_I}{\partial t}+\rho u_i\frac{\partial Z_I}{\partial x_i}=-\frac{\partial J_{I_i}}{\partial x_i}+\dot{m}_I, \tag{2.12}$$

where I indicates the mass concentration, i.e., $I=1,2,...,10$.

2.1.3 Energy Conservation

Conservation of energy is expressed in terms of temperature, T, as

$$\rho C_p\left\{\frac{\partial T}{\partial t}+u_i\frac{\partial T}{\partial x_i}\right\}=-\frac{\partial q_i}{\partial x_i}+Q \tag{2.13}$$

where C_p is the specific heat at constant pressure, q_i is the diffusional heat flux rate, and Q represents volumetric heat sources and sinks, e.g., due to exothermic/endothermic chemical reactions

Fourier's law relates the heat flux rate to the temperature gradient and thermal conductivity,

$$q_i=-k_{ij}\frac{\partial T}{\partial x_j}, \tag{2.14}$$

where k_{ij} is the thermal conductivity tensor. In many cases, the fluid properties are only available as scalar quantities, i.e., the thermal conductivity is considered to be isotropic.

Remark

> In LS-DYNA, the viscosity, thermal conductivity, and mass diffusivities are treated as second-rank tensors even though these properties may only be available as scalar quantities for some fluids. In the limiting case of a scalar material property such as viscosity, the internal code-representation assumes that the viscosity is $\mu_{ij}=\delta_{ij}\hat{\mu}$ where $\hat{\mu}$ is the user-input scalar viscosity.

2.1.4 Boundary and Initial Conditions

The prescription of boundary conditions is based on a flow domain with boundaries that are either physical or implied for the purposes of performing a simulation. A simple flow domain is shown in Figure 2.1 where the boundary of the domain is $\Gamma=\Gamma_1\cup\Gamma_2$.

The momentum equations, Eq. (2.1), are subject to boundary conditions that consist of specified velocity on Γ_1 as in Eq. (2.15), or traction boundary conditions on Γ_2 as in Eq. (2.17).

$$u_i(x_i,t)=\hat{u}_i(x_i,t)\ on\ \Gamma_1 \tag{2.15}$$

In the case of a no-slip and no-penetration boundary, $u_i=0$ is the prescribed velocity boundary condition.

The prescribed traction boundary conditions are

$$\sigma_{ij}n_j=\hat{f}_i(x_i,t)\ on\ \Gamma_2, \tag{2.16}$$

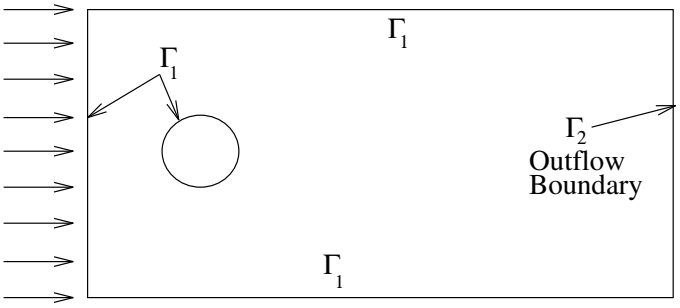


Figure 2.1: Flow domain for conservation equations.

where n_j is the outward normal for the domain boundary, and $\hat{f}_i$ are the components of the prescribed traction. In terms of the pressure and strain-rate, the traction boundary conditions are

$$\left\{-p\delta_{ij} + 2\mu S_{ij}\right\} n_j = \hat{f}_i(x_i,t) \; on \; \Gamma_2. \tag{2.17}$$

The traction and velocity boundary conditions can be mixed. In a two-dimensional sense, mixed boundary conditions can consist of a prescribed normal traction and a tangential velocity. For example, at the outflow boundary in Figure 2.1, a homogeneous normal traction and vertical velocity on Γ_2 constitutes a valid set of mixed boundary conditions. A detailed discussion of boundary conditions for the incompressible Navier-Stokes equations may be found in Gresho and Sani [5].

Turning attention to the species transport equations, boundary conditions for Eq. (2.9) may consist of either a prescribed concentration or a mass flux rate. In the binary mixture example, the prescribed concentration is

$$Z_1(x_i,t) = \hat{Z}_1(x_i,t), \tag{2.18}$$

where $\hat{Z}_1$ is the known value of concentration for species 1. The prescribed mass flux rate is

$$-\rho D_{1_{ij}} \frac{\partial Z_1}{\partial x_j} n_i = \hat{J}_1(x_i,t), \tag{2.19}$$

where $\hat{J}_1(x_i,t)$ is the known mass flux rate through the boundary with normal n_i. The prescribed flux rate may also be specified in terms of a mass transfer coefficient as

$$-\rho D_{1_{ij}} \frac{\partial Z_1}{\partial x_j} n_i = h_{D_\infty}(Z_1 - Z_{1_\infty}), \tag{2.20}$$

where h_{D_∞} is the mass transfer coefficient and Z_{1_∞} is a reference species concentration.

The boundary conditions for the energy equation, Eq. (2.13), consist of a prescribed temperature or heat flux rate. The prescribed temperature is

$$T(x_i,t) = \hat{T}(x_i,t), \tag{2.21}$$

and the prescribed heat flux rate is

$$-k_{ij} \frac{\partial T}{\partial x_j} n_i = \hat{q}(x_i,t), \tag{2.22}$$

where $\hat{q}$ is the known flux rate through the boundary with normal n_i. The heat flux rate may also be prescribed in terms of a heat transfer coefficient,

$$-k_{ij} \frac{\partial T}{\partial x_j} n_i = h(T - T_\infty), \tag{2.23}$$

where h is the heat transfer coefficient, and T_∞ is a reference temperature.

Initial conditions take on the form of prescribed velocity, species and temperature distributions at $t = 0$, i.e.,

$$\begin{aligned} u_i(x_i,0) &= u_i^0(x_i), \\ Z_I(x_i,0) &= Z_I^0(x_i), \\ T(x_i,0) &= T^0(x_i). \end{aligned} \tag{2.24}$$

Remark

For a well-posed incompressible flow problem, the prescribed initial velocity field in Eq. (2.25) must satisfy Eq. (2.25) – (2.26) (see Gresho and Sani[6]). If $\Gamma_2 = 0$ (the null set, i.e., enclosure flows with $n_i u_i$ prescribed on all surfaces), then global mass conservation enters as an additional solvability constraint as shown in Eq. (2.27).

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{2.25}$$

$$n_i u_i(x_i,0) = n_i u_i^0(x_i) \tag{2.26}$$

$$\int_\Gamma n_i u^0 d\Gamma = 0 \tag{2.27}$$

2.2 Conservation Equations – Vector Notation

In the ensuing discussion, the invariant bold-face vector notation of Gibbs is used with **boldface** symbols representing vector/tensor quantities. The reader may refer to Gresho and Sani [5] pp. 357-359 for an outline of notation for the Navier-Stokes equations.

2.2.1 Momentum Conservation

To begin, the conservation of linear momentum is

$$\rho \left\{ \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right\} = \nabla \cdot \sigma + \rho \mathbf{f}, \tag{2.28}$$

where $\mathbf{u} = (u, v, w)$ the velocity, σ is the stress tensor, ρ the mass density, and $\mathbf{f}$ the body force, The body force contribution $\rho\mathbf{f}$ typically accounts for buoyancy forces with $\mathbf{f}$ representing the acceleration due to gravity.

The stress may be written in terms of the fluid pressure and the deviatoric stress tensor as

$$\sigma = -p\mathbf{I} + \tau, \tag{2.29}$$

where p is the pressure, $\mathbf{I}$ is the identity tensor , and τ is the deviatoric stress tensor

A constitutive equation relates the deviatoric stress and the strain rate, e.g.,

$$\tau = 2\mu\mathbf{S}. \tag{2.30}$$

Here, the dynamic viscosity, μ, is a second-rank tensor. Frequently, the fluid viscosity is only available as an isotropic viscosity, in which case, the constitutive relation becomes

$$\tau = 2\mu\mathbf{S}. \tag{2.31}$$

The strain-rate tensor is written in terms of the velocity gradients as

$$\mathbf{S} = \frac{1}{2}\left\{\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right\}. \tag{2.32}$$

2.2.2 Mass Conservation

The mass conservation principle in divergence form is

$$\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\nabla\mathbf{u}) = 0. \tag{2.33}$$

In the incompressible limit, the velocity field is solenoidal,

$$\nabla\cdot\mathbf{u} = 0, \tag{2.34}$$

which implies a mass density transport equation,

$$\frac{\partial\rho}{\partial t} + \mathbf{u}\cdot\nabla\rho = 0. \tag{2.35}$$

For constant density, Eq. (2.35) is neglected with Eq. (2.34) remaining as a constraint on the velocity field.

LS-DYNA provides the capability to transport up to 10 species represented by the mass concentration $Z_1, Z_2, \ldots, Z_{10}$. In order to simplify the presentation, a single mass fraction is presented representing a binary mixture. In order to account for the change in mass concentration, mass conservation applied to the individual species yields for Z_1

$$\rho\frac{\partial Z_1}{\partial t} + \rho\mathbf{u}\cdot\nabla Z_1 = -\nabla\mathbf{J}_1 + \dot{m}_1, \tag{2.36}$$

where $\mathbf{J}_1$ is the diffusional mass flux rate, and $\dot{m}_1$ is a volumetric mass source. The mass source may include the injection of mass concentration from a boundary or the source/sink terms from chemical reactions.

The diffusional mass flux rate is based on Fick's law of diffusion,

$$\mathbf{J}_1 = -\rho\mathbf{D}_1\nabla Z_1, \tag{2.37}$$

where **D** is a tensorial mass diffusivity. Typically mass diffusivities are only available as scalars so that

$$\mathbf{J}_1 = -\rho D_1 \nabla Z_1. \tag{2.38}$$

In the most general form, the species concentration transport equations are

$$\rho \frac{\partial Z_I}{\partial t} + \rho \mathbf{u} \cdot \nabla Z_I = -\nabla \cdot \mathbf{J}_I + \dot{m}_I, \tag{2.39}$$

where I indicates the species mass concentration, i.e., $I = 1, 2, ..., 10$.

2.2.3 Energy Conservation

The conservation of energy is expressed in terms of temperature, T, as

$$\rho C_p \left\{ \frac{\partial T}{\partial t} + \mathbf{u} \nabla T \right\} = -\nabla \cdot \mathbf{q} + Q \tag{2.40}$$

where C_p is the specific heat at constant pressure, **q** is the diffusional heat flux rate, and Q represents volumetric heat sources and sinks, e.g., due to exothermic/endothermic chemical reactions

Fourier's law relates the heat flux rate to the temperature gradient and thermal conductivity,

$$\mathbf{q} = -\mathbf{k} \nabla T, \tag{2.41}$$

where **k** is the thermal conductivity tensor. In many cases, the fluid properties are only available as scalar quantities, i.e., the thermal conductivity is considered to be isotropic.

2.2.4 Boundary and Initial Conditions

The prescription of boundary conditions is based on a flow domain with boundaries that are either physical or implied for the purposes of performing a simulation. A simple flow domain is shown in Figure 2.1 where the boundary of the domain is $\Gamma = \Gamma_1 \cup \Gamma_2$.

The momentum equations, Eq. (2.28), are subject to boundary conditions that consist of specified velocity on Γ_1 as in Eq. (2.42), or traction boundary conditions on Γ_2 as in Eq. (2.44).

$$\mathbf{u}(\mathbf{x}, t) = \hat{\mathbf{u}}(\mathbf{x}, t) \; on \; \Gamma_1 \tag{2.42}$$

In the case of a no-slip and no-penetration boundary, $\mathbf{u} = 0$ is the prescribed velocity boundary condition.

The prescribed traction boundary conditions are

$$\sigma \cdot \mathbf{n} = \hat{\mathbf{f}}(\mathbf{x}, t) \; on \; \Gamma_2, \tag{2.43}$$

where **n** is the outward normal for the domain boundary, and $\hat{\mathbf{f}}$ are the components of the prescribed traction. In terms of the pressure and strain-rate, the traction boundary conditions are

$$\{-p\mathbf{I} + 2\mu\mathbf{S}\} \cdot \mathbf{n} = \hat{\mathbf{f}}(\mathbf{x}, t) \; on \; \Gamma_2. \tag{2.44}$$

The traction and velocity boundary conditions can be mixed. In a two-dimensional sense, mixed boundary conditions can consist of a prescribed normal traction and a tangential velocity. For example, at the outflow boundary in Figure 2.1, a homogeneous normal traction and vertical velocity on Γ_2 constitutes a valid set of mixed boundary conditions. A detailed discussion of boundary conditions for the incompressible Navier-Stokes equations may be found in Gresho and Sani [5].

Turning attention to the species transport equations, boundary conditions for Eq. (2.9) may consist of either a prescribed concentration or a mass flux rate. In the binary mixture example, the prescribed concentration is

$$Z_1(\mathbf{x},t) = \hat{Z}_1(\mathbf{x},t), \tag{2.45}$$

where $\hat{Z}_1$ is the known value of concentration for species 1. The prescribed mass flux rate is

$$-\rho \mathbf{D}_1 \nabla Z_1 \cdot \mathbf{n} = \hat{J}_1(\mathbf{x},t), \tag{2.46}$$

where $\hat{J}_1(x_i,t)$ is the known mass flux rate through the boundary with normal $\mathbf{n}$. The prescribed flux rate may also be specified in terms of a mass transfer coefficient as

$$-\rho \mathbf{D}_1 \nabla Z_1 \cdot \mathbf{n} = h_{D_\infty}(Z_1 - Z_{1_\infty}), \tag{2.47}$$

where h_{D_∞} is the mass transfer coefficient and Z_{1_∞} is a reference species concentration.

The boundary conditions for the energy equation, Eq. (2.13), consist of a prescribed temperature or heat flux rate. The prescribed temperature is

$$T(\mathbf{x},t) = \hat{T}(\mathbf{x},t), \tag{2.48}$$

and the prescribed heat flux rate is

$$-\mathbf{k}\nabla T \cdot \mathbf{n} = \hat{q}(\mathbf{x},t), \tag{2.49}$$

where $\hat{q}$ is the known flux rate through the boundary with normal $\mathbf{n}$. The heat flux rate may also be prescribed in terms of a heat transfer coefficient,

$$-\mathbf{k}\nabla T \cdot \mathbf{n} = h(T - T_\infty), \tag{2.50}$$

where h is the heat transfer coefficient, and T_∞ is a reference temperature.

Initial conditions take on the form of prescribed velocity, species and temperature distributions at $t = 0$, i.e.,

$$\begin{aligned} \mathbf{u}(\mathbf{x},0) &= \mathbf{u}^0(\mathbf{x}), \\ Z_I(\mathbf{x},0) &= Z_I^0(\mathbf{x}), \\ T(\mathbf{x},0) &= T^0(\mathbf{x}). \end{aligned} \tag{2.51}$$

Remark

For a well-posed incompressible flow problem, the prescribed initial velocity field in equation (2.52) must satisfy equations (2.52) and (2.53) (see Gresho and Sani[6]). If $\Gamma_2 = 0$ (the null set, i.e., enclosure flows with $\vec{n} \cdot \mathbf{u}$ prescribed on all surfaces), then global mass conservation enters as an additional solvability constraint as shown in equation (2.54).

$$\nabla \cdot \mathbf{u}^0 = 0 \tag{2.52}$$

$$\mathbf{n} \cdot \mathbf{u}(\mathbf{x}, 0) = \mathbf{n} \cdot \mathbf{u}^o(\mathbf{x}) \tag{2.53}$$

$$\int_\Gamma \mathbf{n} \cdot \mathbf{u}^o d\Gamma = 0 \tag{2.54}$$

Chapter 3

Explicit Time Integration

This chapter presents the spatial discretization and explicit time-integration method for the incompressible Navier-Stokes equations. The spatial discretization is achieved using the $Q1Q0$ element with bilinear support for velocity and piecewise constant support for the pressure in two dimensions. In three dimensions, the velocity support is trilinear with piecewise constant support for pressure. The methods for obtaining the weak-form of the conservation equations are well known and will not be repeated here (see for example, Gresho, et al.[7], Hughes[8], and Zienkiewicz and Taylor[9]). The spatially discrete form of Eq. (2.1) and (2.7) are

$$M\dot{\mathbf{u}} + A(\mathbf{u})\mathbf{u} + K\mathbf{u} + Cp = \mathbf{F}, \tag{3.1}$$

and

$$C^T\mathbf{u} = g, \tag{3.2}$$

where M is the mass matrix, $A(\mathbf{u})$ and K are the the advection and the viscous diffusion operators respectively, $\mathbf{F}$ is the body force, and g accounts for the presence of prescribed velocity boundary conditions. C is the gradient operator, and C^T is the divergence operator. Here, $\mathbf{u}$ and p are understood to be discrete approximations to the continuous velocity and pressure fields. Equations (3.1) and (3.2) constitute a differential-algebraic system of equations that precludes the direct application of time-marching algorithms due to the presence of the discrete incompressibility constraint.

Following Gresho, et al.[1], a consistent, discrete pressure Poisson equation (PPE) is constructed using a row-sum lumped mass matrix, M_L.

$$[C^T M_L^{-1} C]p = C^T M_L^{-1}[\mathbf{F} - K\mathbf{u} - A(\mathbf{u})\mathbf{u}] - \dot{g} \tag{3.3}$$

Here, $\dot{g}$ accounts for time-dependent velocity boundary conditions.

The PPE constitutes an algebraic system of equations that is solved for the element-centered pressure during the time-marching procedure. Figure 3.1 shows the dual, staggered grid associated with the pressure variables. The PPE in Eq. (3.3) incorporates the effect of the essential velocity boundary conditions from Eq. (2.15), and automatically builds in the boundary conditions from Eq. (2.17) – see Gresho, et al. [6].

Equations (3.1) and (3.3) form the basis for a description of the explicit time integration algorithm. It is assumed that the explicit algorithm begins with a given divergence-free velocity field, $\mathbf{u}^0$, that satisfies the essential velocity boundary conditions, and an initial pressure, p^0. To simplify the description of the

explicit algorithm, velocity boundary conditions that are constant in time are assumed making $\dot{g} = 0$ – see Gresho and Sani [10] for additional algorithmic issues for the situation when $\dot{g} \neq 0$. The explicit algorithm proceeds as follows.

1. Calculate the partial acceleration, i.e., acceleration neglecting the pressure gradient, at time level n.

$$\tilde{\mathbf{a}}^n = M_L^{-1}\tilde{\mathbf{F}}^n \tag{3.4}$$

   where

$$\tilde{\mathbf{F}}^n = \mathbf{F}^n - K\mathbf{u}^n - A(\mathbf{u})\mathbf{u}^n \tag{3.5}$$

2. Solve the global PPE for the current pressure field.

$$[C^T M_L^{-1} C]p^n = C^T \tilde{\mathbf{a}}^n \tag{3.6}$$

3. Update the nodal velocities.

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t[\tilde{\mathbf{a}}^n + M_L^{-1} C p^n] \tag{3.7}$$

4. Repeat steps 1-3 until a maximum simulation time limit or maximum number of time steps is reached.

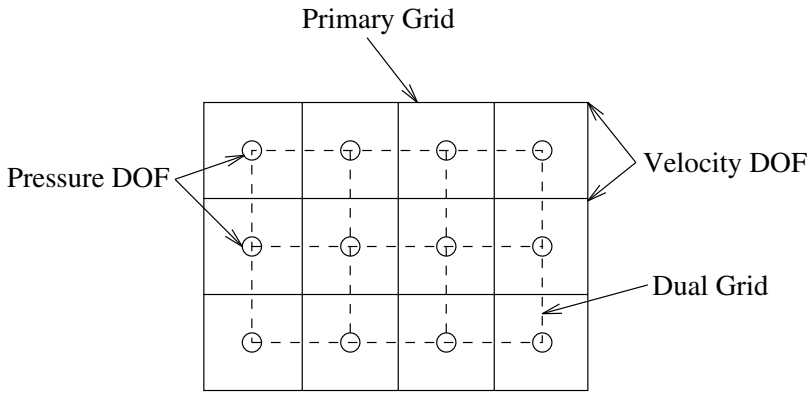


Figure 3.1: Velocity mesh with two degrees-of-freedom (DOF) per node, and the PPE dual grid with one DOF per element.

Remark

In LS-DYNA, the prescribed initial conditions and boundary conditions are tested and, if necessary, a projection to a divergence-free subspace is performed on the initial velocity field, $\mathbf{u}^0$. This guarantees that the flow problem is well-posed, even if the user prescribed initial conditions violate the conditions of Eq. (2.25) - (2.26) presented in Chapter 2.

In practice, the criterion for performing a projection onto a div-free subspace is based upon the RMS divergence error

$$\sqrt{\frac{(C^T\mathbf{u})\cdot(C^T\mathbf{u})}{Nel}} \leq \varepsilon, \tag{3.8}$$

where *Nel* is the number of elements and ε is a user-specified tolerance typically $10^{-10} \leq \varepsilon \leq 10^{-6}$. If the RMS divergence error is greater than the specified tolerance for the initial candidate velocity field, $\hat{\mathbf{u}}^0$, then the PPE problem in Eq. (3.9) is solved for λ, and a mass-consistent projection performed using Eq. (3.10).

$$[C^T M_L^{-1} C]\lambda = C^T \hat{\mathbf{u}}^0 \tag{3.9}$$

$$\mathbf{u}^0 = \hat{\mathbf{u}}^0 - M_L^{-1} C\lambda \tag{3.10}$$

The explicit time-integration algorithm must respect both diffusive and convective stability limits. Although the analytical stability limits for the explicit time integration of the Navier-Stokes equations in multiple dimensions remain intractable[1], approximate stability computations may be performed using local grid metrics.

In an unstructured grid, with variable element size, the calculation of the grid *Re* (Reynolds) and *CFL* (Courant-Freidrichs-Levy) numbers uses the element-local coordinates and centroid velocities. Figure 3.2 shows the canonical element-local node-numbering scheme, coordinate system and centroid velocity for the 2-D and 3-D elements.

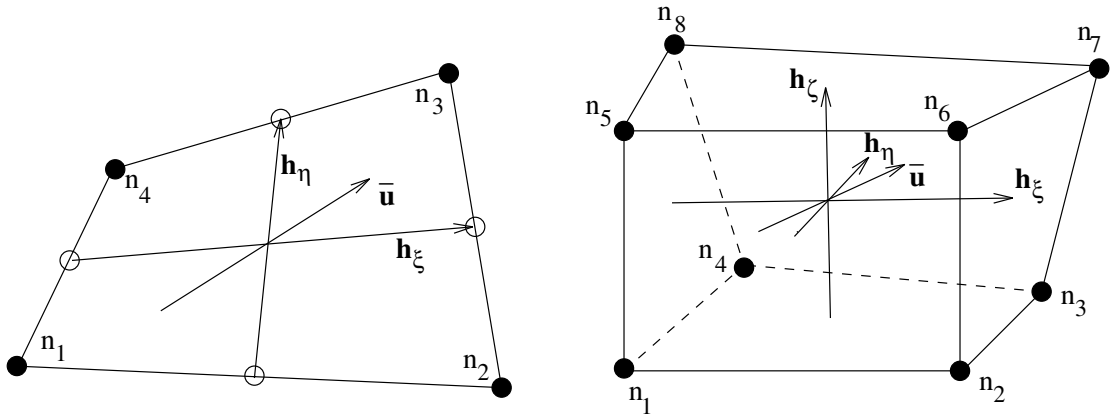


Figure 3.2: Grid parameters for: a) Two-dimensional element with centroid velocity and characteristic element dimensions h_ξ and h_η, b) Three-dimensional element with centroid velocity and characteristic dimensions h_ξ, h_η, and h_ζ.

The grid *Re* and *CFL* numbers are defined as

$$Re_i = \frac{|\overline{\mathbf{u}}\cdot\mathbf{h}_i|}{2\nu} \tag{3.11}$$

$$CFL_i = \frac{|\overline{\mathbf{u}}\cdot\mathbf{h}_i|\Delta t}{||\mathbf{h}_i||^2} \tag{3.12}$$

where $i = \xi, \eta, \zeta$ are the element-local coordinates. The grid Re and CFL numbers rely upon the projection of the centroid velocity onto the element-local coordinate directions that are oriented according to the canonical local node numbering scheme for each element type[11, 12].

In order to use Eq. (3.12) to estimate a stable time step, a unit vector for each element-local coordinate direction is defined as

$$\hat{\mathbf{e}}_i = \frac{\mathbf{h}_i}{||\mathbf{h}_i||} \tag{3.13}$$

where $\hat{\mathbf{e}}_i$ denotes the unit vector for each of the (ξ, η, ζ) coordinate directions. Using the grid Re and the element size, $\mathbf{h}$, the advective-diffusive stability limit becomes

$$\Delta t_i \leq \nu ||\mathbf{h}_i||^2 \left\{ 1 + \sqrt{1 + (Re_i)^2} \right\}^{-1} \tag{3.14}$$

where a minimum over all elements and all element-local coordinate directions establishes a global minimum time-step.

The advective stability limit is established in a similar manner using

$$\Delta t_i \leq \frac{CFL||\mathbf{h}_i||}{|\overline{\mathbf{u}} \cdot \mathbf{h}_i|}. \tag{3.15}$$

The stable time step is based upon the minimum time step derived from either Eq. (3.14) or (3.15). However, for meshes graded to resolve boundary layers, the advective-diffusive stability limit usually dictates the time step.

3.1 Modified Finite Element Formulation

Several ad-hoc modifications are made to the standard Galerkin finite element formulation for the explicit time integration algorithm. These modifications include the use of a row-sum lumped mass matrix, single point Gaussian quadrature, balancing tensor diffusivity (BTD), and hourglass stabilization to damp the spurious zero-energy modes known as *keystone* or *hourglass* modes. A detailed numerical analysis of these modifications is discussed in Gresho, et al.[1]

Before discussing the reduced integration operators, a brief overview of the element matrices associated with Eq. (3.2) and (3.1) is presented. The element level gradient, mass, advection and diffusion operators are presented in Eq. (3.16) – (3.19). Here, N_a is the element shape function, the subscripts a and b range from 1 to the number of nodes per element, $Nnpe$, and $1 \leq i, j \leq Ndim$.

$$C^e_{i_a} = -\int_{\Omega^e} \frac{\partial N_a}{\partial x_i} dV \tag{3.16}$$

$$M^e_{ab} = \int_{\Omega^e} N_a N_b dV \tag{3.17}$$

$$A^e_{ab}(\mathbf{u}) = \int_{\Omega^e} N_a \, u_i \frac{\partial N_b}{\partial x_i} dV \tag{3.18}$$

$$K^e_{ab} = \int_{\Omega^e} \frac{\partial N_a}{\partial x_i} \mu_{ij} \frac{\partial N_b}{\partial x_j} dV \tag{3.19}$$

3.1.1 Reduced-Integration Operators

From an examination of the explicit algorithm, it is clear that the bulk of the computational effort for a discrete time step consists of the formation and assembly of the right-hand-side in Eq. (3.5), and the subsequent PPE solution in Eq. (3.6). The right-hand-side vector is formed element-by-element using a right-to-left matrix-vector multiply with reduced integration operators for the diffusive and advective terms. In this context, reduced integration is considered synonymous with single-point Gaussian quadrature. Using single-point integration, the element area and gradient operators may be written strictly in terms of element-local nodal coordinates.

$$A^e = \frac{1}{2}[x_{31}y_{42} + x_{24}y_{32}] \tag{3.20}$$

$$\begin{aligned} C^e_x &= \frac{1}{2A^e}[y_{24}, y_{31}, -y_{24}, -y_{31}] \\ C^e_y &= \frac{1}{2A^e}[x_{42}, x_{13}, -x_{42}, -x_{13}] \end{aligned} \tag{3.21}$$

In Eq. (3.20) and (3.21), $x_{ab} = x_a - x_b$, where the subscripts a and b identify the local node number and range from 1 to 4 for the two-dimensional bilinear element. The fact that $C_{x3} = -C_{x1}$, and $C_{x4} = -C_{x2}$, permits the storage of only the unique values in the gradient operator at the element level. In two-dimensions, this requires only 4 floating point values per element for C^e_x and C^e_y.

The computation of the element level gradient operators in three-dimensions is somewhat more involved. To begin, the element-local nodal coordinates in the referential domain are defined as follows:

$$\begin{aligned} \xi^T &= [-1, 1, -1, 1, 1, -1, 1, -1] \\ \eta^T &= [-1, 1, 1, -1, -1, 1, 1, -1] \\ \zeta^T &= [-1, -1, -1, -1, 1, 1, 1, 1]. \end{aligned} \tag{3.22}$$

The Jacobian, evaluated at the element centroid, or central Gauss point, is defined in terms of the element-local nodal coordinates $(\mathbf{x}_e, \mathbf{y}_e, \mathbf{z}_e)$ and the referential coordinates, (ξ, η, ζ) in Eq. (3.23).

$$J(0) = \frac{1}{8}\begin{bmatrix} \xi^T\mathbf{x}^e & \xi^T\mathbf{y}^e & \xi^T\mathbf{z}^e \\ \eta^T\mathbf{x}^e & \eta^T\mathbf{y}^e & \eta^T\mathbf{z}^e \\ \zeta^T\mathbf{x}^e & \zeta^T\mathbf{y}^e & \zeta^T\mathbf{z}^e \end{bmatrix} \tag{3.23}$$

The element volume is simply the determinant of the Jacobian in Eq. (3.24). Note that the element volume in three-dimensions may only be computed exactly with single-point integration for elements that are bricks or parallelepipeds.

$$V^e = det(J(0)) \tag{3.24}$$

The computation of the element level gradient operators proceeds by first evaluating the co-factors of the Jacobian, the inverse Jacobian in Eq. (3.25), and the gradient operators as shown in Eq. (3.26). Again, only the unique gradient operators must be stored, i.e., 12 words of storage per element are required for C_x^e, C_y^e, and C_z^e [1].

$$\mathbf{D} = [\mathbf{D}_{ij}] = J(0)^{-1} \tag{3.25}$$

$$\begin{aligned} C_x^e &= \frac{1}{8}[\mathbf{D}_{11}\xi + \mathbf{D}_{12}\eta + \mathbf{D}_{13}\zeta] \\ C_y^e &= \frac{1}{8}[\mathbf{D}_{21}\xi + \mathbf{D}_{22}\eta + \mathbf{D}_{23}\zeta] \\ C_z^e &= \frac{1}{8}[\mathbf{D}_{31}\xi + \mathbf{D}_{32}\eta + \mathbf{D}_{33}\zeta] \end{aligned} \tag{3.26}$$

The computation of the element level mass matrix using one-point quadrature and row-sum lumping yields the operator for 2-D in Eq. (3.27) and the operator for 3-D in Eq. (3.28).

$$M_{ab}^e = \delta_{ab}\frac{A^e}{4} \tag{3.27}$$

$$M_{ab}^e = \delta_{ab}\frac{V^e}{8} \tag{3.28}$$

The direct evaluation of the advection operator in Eq. (3.18) requires an integral of triple products that is very computationally intensive. Therefore, the advection operator is approximated using an *ad-hoc* modification known as the centroid advection velocity. This modification assumes that $\mathbf{u}$ in Eq. (3.18) may be approximated by

$$\overline{\mathbf{u}} = \sum_{a=1}^{Nnpe} N_a(\mathbf{0})\mathbf{u}_a \tag{3.29}$$

where $N_a(\mathbf{0})$ indicates evaluation of the shape functions at the origin of the referential coordinate system.

The application of single-point integration further simplifies the advection operator. In two-dimensions, α_1 and α_2 in Eq. (3.30) are used to form the advection operators in Eq. (3.31) where $u_{ab} = u_a - u_b$.

$$\begin{aligned} \alpha_1 &= \overline{u}C_{x1} + \overline{v}C_{y1} \\ \alpha_2 &= \overline{u}C_{x2} + \overline{v}C_{y2} \end{aligned} \tag{3.30}$$

$$\begin{aligned} A^e(\overline{\mathbf{u}})u^e &= [1,1,1,1]^T \frac{A^e}{4}[\alpha_1 u_{13} + \alpha_2 u_{24}] \\ A^e(\overline{\mathbf{u}})v^e &= [1,1,1,1]^T \frac{A^e}{4}[\alpha_1 v_{13} + \alpha_2 v_{24}] \end{aligned} \tag{3.31}$$

For the evaluation of the three-dimensional advection operators, $\beta_1 - \beta_4$ are defined as

$$\begin{aligned} \beta_1 &= \overline{u}C_{x1} + \overline{v}C_{y1} + \overline{w}C_{z1} \\ \beta_2 &= \overline{u}C_{x2} + \overline{v}C_{y2} + \overline{w}C_{z2} \\ \beta_3 &= \overline{u}C_{x3} + \overline{v}C_{y3} + \overline{w}C_{z3} \\ \beta_4 &= \overline{u}C_{x4} + \overline{v}C_{y4} + \overline{w}C_{z4}, \end{aligned} \tag{3.32}$$

and used in the element-level advection operators.

$$\begin{aligned} A^e(\overline{\mathbf{u}})u^e &= [1,1,1,1,1,1,1,1]^T \frac{V^e}{8}[\beta_1 u_{17} + \beta_2 u_{28} + \beta_3 u_{35} + \beta_4 u_{46}] \\ A^e(\overline{\mathbf{u}})v^e &= [1,1,1,1,1,1,1,1]^T \frac{V^e}{8}[\beta_1 v_{17} + \beta_2 v_{28} + \beta_3 v_{35} + \beta_4 v_{46}] \\ A^e(\overline{\mathbf{u}})w^e &= [1,1,1,1,1,1,1,1]^T \frac{V^e}{8}[\beta_1 w_{17} + \beta_2 w_{28} + \beta_3 w_{35} + \beta_4 w_{46}] \end{aligned} \tag{3.33}$$

The single-point diffusion operator may be stated simply as

$$K^e_{ab} = C_{i_a}\, \hat{\mu}_{ij} C_{j_b} V^e \tag{3.34}$$

where $\hat{\mu}_{ij}$ represents a tensorial viscosity. Here, i and j range from 1 to the number of spatial dimensions while a, b range from 1 to the number of nodes per element. Generation of the diffusion operator in Eq. (3.34) using single point integration leads to rank deficiency of the element level operator. The presence of an improper singular mode in the element level operator may also lead to singularity in the assembled global operator. In two-dimensions, there is only one improper singular mode, while in three-dimensions, there are four singular modes. These modes are commonly referred to as *hourglass* modes, and when excited in a numerical solution, they can remain undamped and pollute the field solution.

A detailed discussion of the hourglass stabilization methods used in LS-DYNA is beyond the scope of this chapter. However, for the sake of completeness, a brief overview of the so-called *h-stabilization* is presented. The term, h-stabilization, derives from the fact that the outer product of the element hourglass vectors is used to form the stabilization operator.

In two-dimensions, the single hourglass mode is

$$\Gamma^T = [1,-1,1,-1]. \tag{3.35}$$

This mode, when excited, may be detected visually as a *w-mode* in isometric plots of the field variable from a numerical solution. Following Goudreau and Hallquist[13] and Gresho, et al.[1], the element level stabilization operator is formed using the outer-product of the hourglass vector.

$$H^e_{ab} = \varepsilon_{hg}\mu A^e \Gamma^T_a \Gamma_b \tag{3.36}$$

ε_{hg} is a non-dimensional parameter that, in practice, is unity by default for outer-product stabilization (see Gresho, et al.[1]).

In three-dimensions, the four hourglass vectors are

$$\begin{aligned}
\Gamma^T_1 &= [1,1,-1,-1,-1,-1,1,1] \\
\Gamma^T_2 &= [1,-1,-1,1,-1,1,1,-1] \\
\Gamma^T_3 &= [1,-1,1,-1,1,-1,1,-1] \\
\Gamma^T_4 &= [-1,1,-1,1,1,1,-1,1,-1].
\end{aligned} \tag{3.37}$$

The resulting 3-D hourglass stabilization operator is

$$H^e_{ab} = \varepsilon^e_{hg}\mu[\Gamma_1\ \Gamma_2\ \Gamma_3\ \Gamma_4] \begin{bmatrix} C_1 & & & \\ & C_2 & & \\ & & C_3 & \\ & & & C_4 \end{bmatrix} \begin{Bmatrix} \Gamma_1 \\ \Gamma_2 \\ \Gamma_3 \\ \Gamma_4 \end{Bmatrix} \tag{3.38}$$

where $\varepsilon^e_{hg} = 1.0$, $C_1 = C_2 = C_3 = C_4 = \overline{h}\sqrt[3]{V^e}$, $\overline{h} = (\sqrt[3]{V_{max}} - \sqrt[3]{V_{min}})/2$.

For the reduced integration element, γ-stabilization, see Belytschko, et al[14, 15, 16], has also been investigated. γ-stabilization refers to the γ-vectors constructed from the hourglass modes for stabilization. While γ-stabilization is perhaps more robust than h-stabilization, this type of hourglass control also requires more operations and storage. It is the author's experience that it is relatively more difficult to excite the hourglass modes in an Eulerian computation than in a Lagrangian computation, e.g., a DYNA3D[17] simulation. However, γ-stabilization still requires fewer operations and less storage than the fully integrated two-dimensional bilinear element.

In three-dimensions, this is not the case. Table 3.1 shows the memory requirements and operations counts for a matrix-vector multiply (Ku) for a variety of element formulations. In 2-D, γ-stabilization requires nearly the same storage as the fully integrated element stored in either a compact, symmetric, element form, or in a global row-compressed form. However, this element requires 9 more operations to achieve the matrix-vector multiply when compared to the element-by-element matrix vector multiply with 2x2 quadrature. In 3-D, γ-stabilization is about 3 times more expensive to perform than the corresponding global row-compressed matrix-vector multiply.

There is one final modification to the finite element formulation that derives from the explicit treatment of the advective terms. For advection dominated flows, it is well known that the use of a backward-Euler treatment of the advective terms introduces excessive diffusion. Similarly, Gresho, et al.[1] have shown

Table 3.1: Memory requirements and operations counts for a matrix-vector multiply for various element integration rules, stabilization operators, and storage schemes. ($Nel = number\ of\ elements$, $Nnp = number\ of\ nodes$, $EBE = element-by-element$)

Dimension	Element Formulation	Storage	+	*	Total Op.
2-D	1-pt., EBE	$4Nel$	$12Nel$	$16Nel$	$28Nel$
2-D	1-pt., h-stabilization, EBE	$4Nel$	$22Nel$	$18Nel$	$40Nel$
2-D	1-pt., γ-stabilization, EBE	$9Nel$	$19Nel$	$25Nel$	$44Nel$
2-D	2x2 Quadrature, EBE	$10Nel$	$16Nel$	$16Nel$	$32Nel$
2-D	2x2, ITPACKV[18]	$9Nnp$	$8Nnp$	$9Nnp$	$17Nnp$
3-D	1-pt., EBE	$12Nel$	$29Nel$	$28Nel$	$57Nel$
3-D	1-pt., h-stabilization, EBE	$12Nel$	$69Nel$	$46Nel$	$115Nel$
3-D	1-pt., γ-stabilization, EBE	$45Nel$	$61Nel$	$100Nel$	$161Nel$
3-D	2x2x2 Quadrature, EBE	$36Nel$	$64Nel$	$64Nel$	$128Nel$
3-D	2x2x2, ITPACKV[18]	$27Nnp$	$26Nnp$	$27Nnp$	$53Nnp$

that forward-Euler treatment of the advective terms results in negative diffusivity, or an under-diffusive scheme. In order to remedy this problem, balancing-tensor diffusivity (BTD), derived from a Taylor series analysis to exactly balance the diffusivity deficit, is adopted. In the one-point quadrature element, the BTD term is simply added to the kinematic viscosity in Eq. (3.39) to form the tensorial diffusivity used in Eq. (3.34).

$$\hat{\mu}_{ij} = \mu_{ij} + \rho \frac{\Delta t}{2} u_i u_j \tag{3.39}$$

In summary, the modifications made to the standard finite element formulation include the use of single-point integration, a row-sum lumped mass matrix, hourglass stabilization, and balancing tensor diffusivity. The benefits promised by one-point integration are tremendous in computational fluid dynamics problems because of the requisite mesh sizes for interesting problems and the concomitant memory requirements.

The reduction from 8 quadrature points to 1 in three dimensions reduces the computational load by a factor of about 6 to 7 and reduces memory requirements by over a factor of 2 for the basic gradient operators. Neglecting the storage costs associated with the PPE, the total storage requirements for the explicit algorithm is 60 words per 3-D element. Further, it has been demonstrated that the convergence rate of the one-point elements is comparable to the fully integrated elements at a fraction of the computational cost, see Liu[16]. With the element formulation defined, attention is now turned to the parallel aspects of the explicit algorithm when a domain-decomposition message-passing paradigm is used.

3.2 The Domain-Decomposition Message-Passing Paradigm

This section describes the domain-decomposition message-passing (DDMP) implementation of the explicit time integration algorithm for the the incompressible Navier-Stokes. A brief overview of domain-decomposition and the associated sub-domain to processor mapping is discussed first. Next, the parallel right-hand-side assembly procedure is presented. The parallel assembly procedure then sets the stage for a discussion of the parallel iterative solution methods applied to the PPE.

3.2.1 Domain-Decomposition

Domain-decomposition is the process of sub-dividing the spatial domain into sub-domains that can be assigned to individual processors for the subsequent solution process. There has been a great deal of work on the problem of spatial decomposition for unstructured grids over the last several years[19, 20, 21, 22, 23, 24, 25, 26]. In general, the requirements for mesh decomposition is that the sub-domains be generated in such a way that the computational load is uniformly distributed across the available processors, and that the inter-processor communication is minimized. LS-DYNA uses recursive coordinate bisection for the decomposition task. Details on the use of decomposition tool and decomposition options may be found in the LS-DYNA User's Manual [27].

In order to exploit the finite element assembly process[8] for parallelization, the dual graph of the finite element mesh, i.e., the connectivity of the dual grid shown in Figure 3.1, is used to perform a non-overlapping element-based domain decomposition. Note that the dual grid corresponds to the grid associated with the element centered pressure variables in the $Q1Q0$ element. Implicit in this choice of a domain decomposition strategy is the idea that elements are uniquely assigned to processors while the nodes at the sub-domain interfaces are stored redundantly in multiple processors. Figure 3.3 illustrates a non-overlapping domain decomposition for a simple two-dimensional vortex-shedding mesh partitioned for four processors. After obtaining the domain decomposition, the assignment of nodes, boundary conditions, and materials to individual processors is performed internally before parallel execution begins.

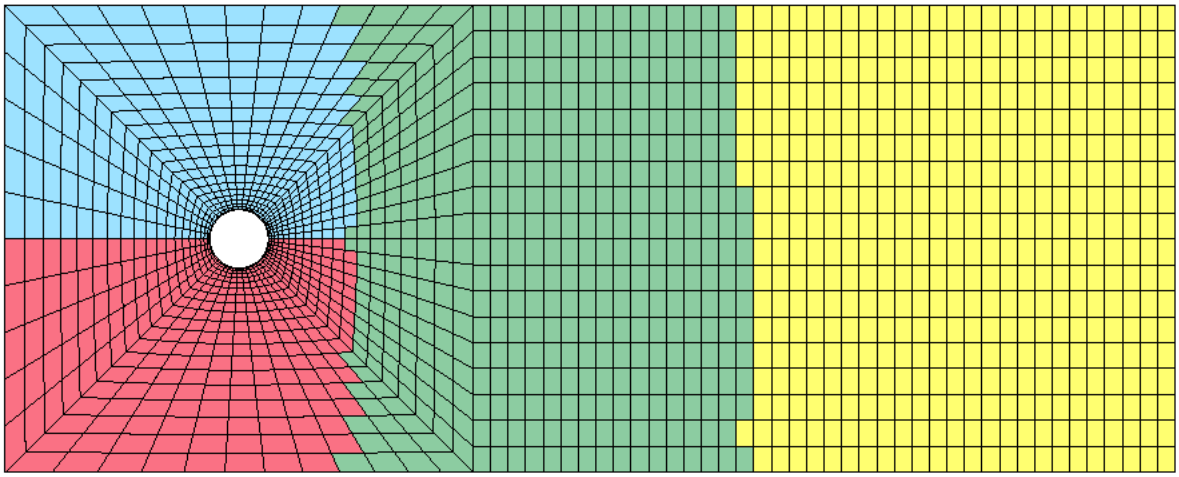


Figure 3.3: Four processor spatial domain decomposition.

3.2.2 Parallel Assembly via Message Passing

The finite element assembly procedure is an integral part of any finite element code and consists of a global gather operation of nodal quantities, an add operation, and a subsequent scatter back to the global memory locations. A complete description of the sequential assembly algorithm may be found in Hughes[8]. The assembly procedure is used to both form global coefficient matrices, right-hand-side vectors, and matrix-vector products in an element-by-element sense. In the case of the explicit time integration algorithm, the emphasis is upon the assembly of the element level contributions to the global right-hand-side vector

$$\hat{\mathbf{F}} = A_{e=1}^{Nel}\{\mathbf{f}_e^n - K_e\mathbf{u}_e^n - A_e(\overline{\mathbf{u}})\mathbf{u}_e^n\} \tag{3.40}$$

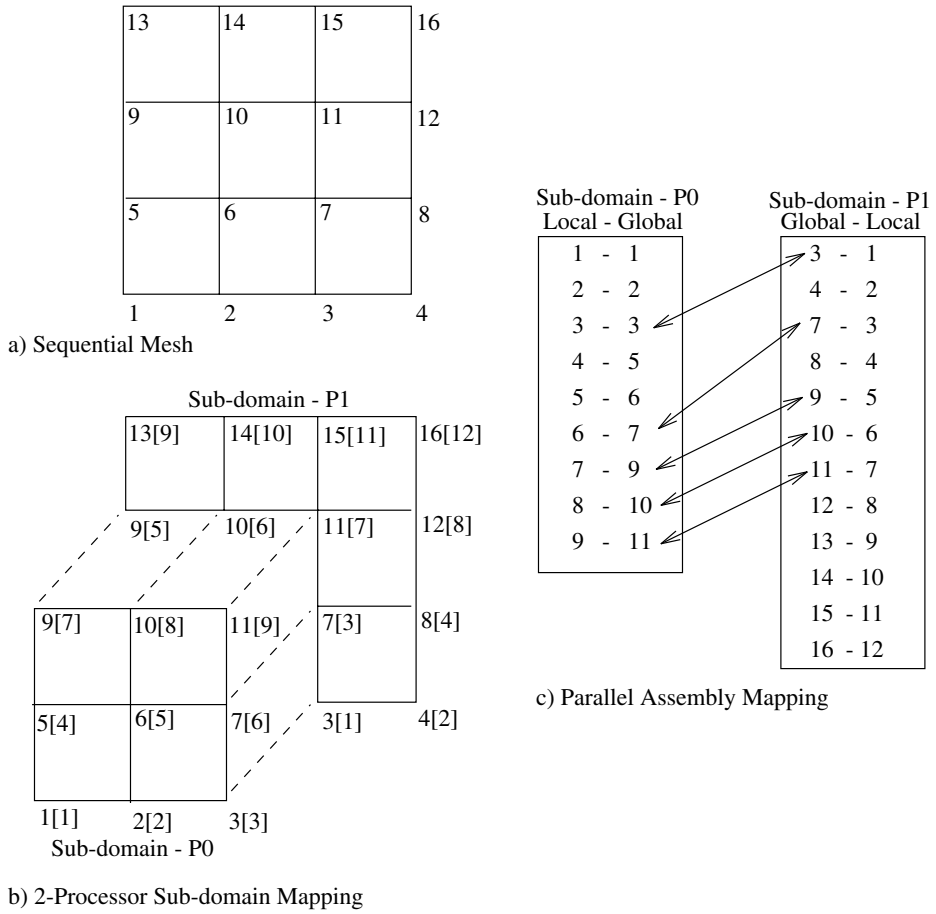


Figure 3.4: Parallel assembly procedure: a) Sequential Mesh, b) 2-Processor sub-domain mapping, c) Parallel assembly mapping.

where A is the assembly operator. Here, the diffusive and advective contributions are computed at the element-level using un-rolled right-to-left matrix-vector multiplies.

In order to exploit both register-to-register vector and cache-based processors, data-independent groups of elements are identified. This not only permits the vectorization of the assembly process, but also maximizes the number of element level operations with respect to the number of load-store operations. For cache based architectures, this permits all of the element data in a data-independent group to be loaded into cache once for the element-level operations, e.g., right-to-left matrix-vector multiplies. Thus, the assembly process using the vector/cache blocks proceeds block-by-block with all element level operations for a block being performed before completing the "add-scatter" portion of the gather-add-scatter assembly procedure.

The parallel assembly procedure may be viewed as a generalized form of the finite element assembly algorithm. However, inter-processor communication is an inherent part of the parallel assembly. As an example of a two-processor assembly, consider the sequential mesh and the assignment of the global nodes to two processors as shown in Figure 3.4. In Figure 3.4b, the local node numbers are enclosed in brackets to the right of the global node number (global node 1 in sub-domain $P0$ is local node $[1]$). The sub-domain assignment of the global nodes is the consequence of the unique assignment of elements to processors, and reveals the existence of the nodes at the sub-domain boundaries on multiple processors.

Figure 3.4b shows the global inter-processor assembly of the sub-domain boundary nodes. Figure 3.4c

illustrates the use of the local-to-global mapping required for the gather-add-scatter operation and the arrows between global node numbers identify a send-receive pair. Thus, the parallel assembly procedure induces communication in the form of a gather-send-receive-add-scatter process. The parallel right-hand-side assembly for Eq. (3.5) may be viewed as a generalized assembly with off-processor communication where first the on-process vector/cache blocked assembly is performed according to Eq. (3.41), and then the assembly at the sub-domain boundaries is performed according to Eq. (3.42).

$$\hat{\mathbf{F}}^p = A_{e=1}^{Nel}\{\mathbf{f}_e^n - K_e\mathbf{u}_e^n - A_e(\overline{\mathbf{u}})\mathbf{u}_e^n\} \tag{3.41}$$

$$\hat{\mathbf{F}} = A_{p=0}^{Np-1}\{\hat{\mathbf{F}}^p\} \tag{3.42}$$

There are several things to note about the parallel assembly procedure. First, the parallel assembly is simply a generalization of the sequential assembly procedure that includes inter-processor communication. Second, the algorithm only requires the communication of nodal data at the edges of adjacent sub-domains. Therefore, as the problem size increases, the communication overhead scales with number of surface nodes associated with sub-domain boundaries. Finally, this algorithm permits the implementation of vector-valued messages in order to avoid start-up issues associated with short messages, i.e., for the assembly shown in Eq. (3.42), the message length is proportional to the number of sub-domain boundary nodes, and the number of degrees-of-freedom per node. Finally, the use of non-overlapping grids implies that nodes in the finite element mesh that lie on sub-domain boundaries are stored in multiple processors as shown in Figure 3.4b. In contrast, over-lapping sub-domains would require the redundant storage of all the data associated with elements at the sub-domain boundaries.

3.2.3 The Parallel Explicit Algorithm

In this section, the issue of solving the PPE in parallel will not be addressed so that attention may be focused upon the solution process for the nodal variables. The DDMP version of the explicit time integration algorithm proceeds as follows.

1. Calculate the partial acceleration, i.e., acceleration neglecting the pressure gradient, at time level n.

   i. All processors calculate the processor-local right-hand-side terms neglecting the pressure gradient according to Eq. (3.41).

   ii. Perform the inter-processor finite element assembly according to Eq. (3.42). In this step, processors that share a common sub-domain boundary exchange messages containing partially assembled right-hand side contributions, and accumulate the fully-summed terms at the sub-domain boundaries.

   iii. All processors compute the processor-local partial acceleration using a pre-calculated lumped mass matrix, i.e., the mass matrix has been fully summed for the nodes at the sub-domain boundaries.

$$\tilde{\mathbf{a}}^n = M_L^{-1}\hat{\mathbf{F}} \tag{3.43}$$

2. Solve the global PPE problem for the current pressure field.

$$[C^T M_L^{-1} C] P^n = C^T \tilde{\mathbf{a}}^n, \tag{3.44}$$

where $\dot{g}^n = 0$ has been used for simplicity.

3. Update the nodal velocities.

i. The computation of the pressure gradient term in Eq. (3.45) consists of the parallel on-processor computation of the discrete pressure gradient, followed by an inter-processor assembly.

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t[\tilde{\mathbf{a}}^n + M_L^{-1}\{A_{p=0}^{Np-1}(CP^n)\}] \tag{3.45}$$

4. Repeat steps 1-3 until a maximum simulation time limit or maximum number of time steps is reached.

Remark

For the explicit solution algorithm, the global row-sum lumped mass is accumulated at all nodes in the finite element mesh during the initialization phase. This requires the nodal assembly of the partial nodal mass at nodes on sub-domain boundaries as shown in Figure 3.4.

3.2.4 Communication Costs

This section outlines the communications costs associated with the explicit time integration algorithm. The communications costs may be broken down into the cost per time step for the momentum equations, and the cost per time step for solving the PPE. To begin, N_Γ is the number of nodes on the boundary of a sub-domain. In two-dimensions, the average number of nodes communicated to an adjacent processor is $N_\Gamma/4$, and in three-dimensions the average is $N_\Gamma/6$. Assuming that there are 8 bytes per floating point word, then the total number of bytes per processor to be communicated via a send is

$$N_c = 8 N_\Gamma N_{DOF}, \tag{3.46}$$

where N_{DOF} is the number of degrees-of-freedom per node.

The communication cost for a send operation may be broken into three parts: the time to initiate the message passing, $t_{startup}$, the cost per packet, t_{packet}, and the transmission time, $t_{transmit}$. Thus, the time to send a message is

$$t_{send} = t_{startup} + \frac{N_c}{N_{packet}} t_{packet} + N_c t_{transmit}, \tag{3.47}$$

where N_{packet} is the number of bytes per packet.

From Eq. (3.46) and (3.47), it is clear that the number of adjacent sub-domains determines the total startup time for message passing. Next, the communication cost per time step is estimated for the explicit algorithm. For the momentum equations, there are two primary messaging steps. The first occurs during the distributed assembly of the right-hand-side in Eq. (3.41). The second occurs during the assembly of the pressure-gradient in Eq. (3.45). Thus, there are 2 vector-valued messaging steps. For the solution

of the PPE, there is 1 vector-valued messaging step per iteration, where N_{IT} iterations are required in the conjugate gradient solution. From this, it is possible to estimate the communication cost per time step for the explicit algorithm on a per processor basis as

$$C_{step} = 8(2 + N_{IT})N_{DOF}N_{\Gamma}t_{send} \tag{3.48}$$

Chapter 4

The Projection Method

The solution of the time-dependent incompressible Navier-Stokes equations poses several algorithmic issues due to the div-free constraint, and the concomitant spatial and temporal resolution required to perform time-accurate solutions particularly where complex geometry is involved. Although fully-coupled solution strategies are available, the cost of such methods is generally considered prohibitive for time-dependent simulations where high-resolution grids are required. The application of projection methods provides a computationally efficient alternative to fully-coupled solution methods.

A detailed review of projection methods is beyond the scope of this chapter, but a partial list of relevant work is provided. Projection methods, also commonly referred to as fractional-step, pressure correction methods, or Chorin's method [28] have grown in popularity over the past 10 years due to the relative ease of implementation and computational performance. This is reflected by the volume of work published on the development of second-order accurate projection methods, see for example van Kan [29], Bell, et al. [30], Gresho, et al. [3, 4, 31, 32], Almgren, et al. [33, 34, 35], Rider [36, 37, 38, 39], Minion [40], Guermond and Quartapelle [41], Puckett, et al. [42], Sussman, et al. [43], and Knio, et al. [44]. The numerical performance of projection methods has been considered by Brown and Minion [45, 46], Wetton [47], Guermond [48, 49], Guermond and Quartapelle [50, 51], and Almgren et al. [52].

As background, a brief review of Chorin's original projection method is presented before proceeding with the finite element form of the projection algorithm. The vector form of the momentum equations may be written as

$$\rho \frac{\partial \mathbf{u}}{\partial t} + \nabla p = F(\mathbf{u}), \tag{4.1}$$

where for a constant viscosity,

$$F(\mathbf{u}) = \mathbf{f} + \mu \nabla^2 \mathbf{u} - \rho \mathbf{u} \cdot \nabla \mathbf{u}. \tag{4.2}$$

Now, $F(\mathbf{u})$ may be decomposed into a div-free and curl-free part where the div-free part is

$$\nabla \cdot \left\{ \frac{\partial \mathbf{u}}{\partial t} \right\} = 0, \tag{4.3}$$

and the curl-free part is

$$\nabla \times \nabla p = 0. \tag{4.4}$$

Discretizing in space and time, the decomposition, neglecting the contribution of the pressure gradient,

yields

$$\rho\frac{(\tilde{\mathbf{u}}^{n+1}-\mathbf{u}^n)}{\Delta t} = F^h(\mathbf{u}), \tag{4.5}$$

where $F^h(\mathbf{u})$ is the spatially discrete analogue of F in Eq. (4.2), and $\tilde{\mathbf{u}}^{n+1}$ is an approximate discrete velocity field at time $n+1$. Note that the discrete divergence of $\tilde{\mathbf{u}}^{n+1}$ is generally not zero, i.e. $G^T\tilde{\mathbf{u}}^{n+1} \neq 0$ where G^T is the discrete divergence operator. The functional dependence of F^h upon the discrete velocity, $\mathbf{u}$, depends upon whether the algorithm is implicit, explicit, or semi-implicit. The dependence on pressure, or rather pressure gradient, is

$$\frac{(\tilde{\mathbf{u}}^{n+1}-\mathbf{u}^n)}{\Delta t} = \frac{(\mathbf{u}^{n+1}-\mathbf{u}^n)}{\Delta t} - \frac{1}{\rho}Gp^{n+1}, \tag{4.6}$$

where G is the discrete gradient operator, and G^T is the discrete divergence operator. Applying the discrete divergence operator to Eq. (4.6) yields a Poisson equation for the pressure at time level $n+1$,

$$G^T\frac{1}{\rho}Gp^{n+1} = \frac{1}{\Delta t}G^T\tilde{\mathbf{u}}^{n+1}. \tag{4.7}$$

By eliminating the velocity at time level n, Eq. (4.6) yields a relationship for the projected div-free velocity field.

$$\mathbf{u}^{n+1} = \tilde{\mathbf{u}}^{n+1} - \frac{1}{\rho}Gp^{n+1}. \tag{4.8}$$

Projection Properties

The philosophy behind projection algorithms is to provide a legitimate way to decouple the pressure and velocity fields in the hope of providing an efficient computational method for transient, incompressible flow simulations. Thus, given an approximate, non-solenoidal velocity field, $\tilde{\mathbf{u}}$, $F(\tilde{\mathbf{u}})$ may be projected onto a divergence-free subspace such that

$$\rho\frac{\partial \mathbf{u}}{\partial t} = P(F(\tilde{\mathbf{u}})), \tag{4.9}$$

and

$$\nabla p = Q(F(\tilde{\mathbf{u}})). \tag{4.10}$$

Here, P and Q are the projection operators, and they have the following properties. P projects a velocity vector onto a div-free subspace, and Q projects a vector into a curl-free subspace. Both P and Q are idempotent, i.e., $P = P^2$ and $Q = Q^2$. Therefore, repeated application of the projection operators does not continue to modify the projected results. The projection operators are orthogonal, and commute, i.e., $PQ = QP = 0$.

The explicit forms of the continuous projection operators are

$$P(\cdot) = \left\{I - \nabla(\nabla^2)^{-1}\nabla\cdot\right\}(\cdot), \tag{4.11}$$

and

$$Q(\cdot) = I - P = \nabla(\nabla^2)^{-1}\nabla\cdot(\cdot). \tag{4.12}$$

It should be noted that P, and Q have *built-in* all the appropriate physical boundary conditions. Further, The eigenvalues of P and Q are either 0 or 1 so that the projections are norm-reducing.

In practice, the action of the projection, P, is to remove the part of the approximate velocity field that is not div-free, i.e., $\mathbf{u} = P(\tilde{\mathbf{u}})$. In effect, the projection is achieved by decomposing the velocity field into div-free and curl-free components using a Helmholtz decomposition. The decomposition may be written as

$$\tilde{\mathbf{u}} = \mathbf{u} + \nabla\lambda, \tag{4.13}$$

where $\tilde{\mathbf{u}}$ is a non-solenoidal velocity field, $\mathbf{u}$ is its div-free counterpart, and $\nabla\lambda$ is the curl-free component, i.e., $\nabla \times \nabla\lambda = 0$.

4.1 Semi-Implicit Projection Method

In LS-DYNA, the Projection-2 (P2) method identified by Gresho [3] forms the starting point for a discussion of the finite element projection algorithms. Before proceeding with a description of the P2 algorithm, the semi-discrete Navier-Stokes equations are presented. The spatial discretization of the conservation equations is achieved using the $Q1Q0$ element with bilinear support for velocity and piecewise constant support for the pressure in two dimensions. In three dimensions, the velocity support is trilinear with piecewise constant support for pressure. The methods for obtaining the weak-form of the conservation equations are well known and will not be repeated here (see for example, Gresho, et al.[7], Hughes[8], and Zienkiewicz and Taylor[9]). The spatially discrete form of momentum conservation, Eq. (2.1), and the divergence constraint, Eq. (2.7), are

$$M\dot{\mathbf{u}} + A(\mathbf{u})\mathbf{u} + K\mathbf{u} + MM_L^{-1}Cp = \mathbf{F}, \tag{4.14}$$

$$C^T\mathbf{u} = g, \tag{4.15}$$

where M is the unit mass matrix, $A(\mathbf{u})$ and K are the advection and the viscous diffusion operators respectively, $\mathbf{F}$ is the body force, and g accounts for the presence of prescribed velocity boundary conditions. C is the gradient operator, and C^T is the divergence operator, i.e., C and C^T are nearly the discrete finite element analogues of G and G^T discussed above. In order to simplify the nomenclature, $\mathbf{u}$ and p are understood to be discrete approximations to the continuous velocity and pressure fields.

Following the development of the projection method introduced above,

$$M\left(\frac{\tilde{\mathbf{u}}^{n+1} - \mathbf{u}^n}{\Delta t}\right) = F^h(\mathbf{u}, p), \tag{4.16}$$

where $F^h(\mathbf{u}, p)$ may involve an explicit or implicit dependence upon $\mathbf{u}$ and is understood to be a discrete vector quantity. From this, it is clear that the intermediate velocity, $\tilde{\mathbf{u}}^{n+1}$, corresponds to an approximate velocity field that has not yet felt the influence of the current pressure field, and therefore is not necessarily solenoidal.

Incorporating the pressure gradient contribution, the approximation to $F^h(\mathbf{u}, p)$ is,

$$M\left(\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t}\right) = M\left(\frac{\tilde{\mathbf{u}}^{n+1} - \mathbf{u}^n}{\Delta t}\right) + MM_L^{-1}Cp^{n+1}. \tag{4.17}$$

Eliminating the velocity at time level n, yields the discrete statement of the Helmholtz decomposition with the concomitant div-free constraint, $C^T\mathbf{u}^{n+1} = g^{n+1}$, viz.,

$$\begin{bmatrix} M_L & C \\ C^T & 0 \end{bmatrix} \begin{Bmatrix} \mathbf{u}^{n+1} \\ \lambda \end{Bmatrix} = \begin{Bmatrix} M_L\tilde{\mathbf{u}}^{n+1} \\ g^{n+1} \end{Bmatrix}. \tag{4.18}$$

Here, M_L is the row-sum lumped, i.e., diagonalized, mass matrix, and $\lambda = \Delta t(p^{n+1} - p^n)$

The system of equations in Eq. (4.18) is analogous to those introduced by Chorin [28]. Although Chorin suggested using a method of successive substitutions to obtain the velocity and pressure fields corresponding to the div-free state, the solution of Eq. (4.18) via a gradient-based iterative method and the solution of the Schur complement of Eq. (4.18) has proven more efficient. The Schur complement of the *projection* operator may be formed explicitly from Eq. (4.18) yielding

$$[C^T M_L^{-1} C]\lambda = C^T\tilde{\mathbf{u}}^{n+1} - g^{n+1}, \tag{4.19}$$

where M_L is the diagonalized, i.e., row-sum lumped mass matrix and includes the prescription of essential velocity boundary conditions. Note that the term *projection* operator is used loosely here to describe Eq. (4.18) since its solution yields a div-free velocity field and the corresponding Lagrange multiplier. However, this operator is not to be confused with the continuous projection operators, P and Q.

Eq. (4.19) is the consistent, discrete form of the elliptic operator for the projection algorithm. It represents an algebraic system of equations that is solved for the element-centered Lagrange multiplier during the time-marching procedure. Figure 3.1 shows the dual, staggered grid where λ and P are centered. The pressure-Poisson operator in Eq. (4.19) incorporates the effect of the essential velocity boundary conditions, and automatically builds in the boundary conditions from Eq. (2.15) and (2.17) – see Gresho, et al. [4].

The projection algorithm proceeds as follows. Given a div-free velocity, $\mathbf{u}^n$, and its corresponding pressure field, p^n, solve the momentum equations for an approximate velocity field at $n+1$.

1. Calculate the approximate velocity field $\tilde{\mathbf{u}}^{n+1}$

$$\begin{aligned}[M + \Delta t\theta_K K]\tilde{\mathbf{u}}^{n+1} = \\ [M - \Delta t(1-\theta_K)K]\mathbf{u}^n + \\ \Delta t\{\theta_F\mathbf{F}^{n+1} + (1-\theta_F)\mathbf{F}^n - A(\mathbf{u}^n)\mathbf{u}^n - MM_L^{-1}Cp^n\}.\end{aligned} \tag{4.20}$$

   Here, θ_K and θ_F control the time-weighting applied to the viscous and body force terms with $0 \leq \theta_K, \theta_F \leq 1$. For $\theta_K = 1$, the viscous terms are treated explicitly, while for $\theta_K = 0$, the viscous terms are treated implicitly, i.e., via backward-Euler, and for $\theta_K = 1/2$, the viscous treatment corresponds to a Crank-Nicolson integrator.

2. Given the approximate velocity, $\tilde{\mathbf{u}}^{n+1}$, solve Eq. (4.19) for λ.

3. Project the approximate velocity to a div-free subspace.

$$\mathbf{u}^{n+1} = \tilde{\mathbf{u}}^{n+1} - M_L^{-1}C\lambda \tag{4.21}$$

4. After the velocity update, an updated pressure at time level $n+1$ is obtained via

$$p^{n+1} = p^n + \frac{\lambda}{\Delta t}. \tag{4.22}$$

Remark

In the projection algorithm, the start-up procedure to obtain the pressure, p^0, proceeds as follows.

1. Calculate the partial acceleration, i.e., acceleration neglecting the pressure gradient, at time level n by solving the mass-matrix problem.

$$M\tilde{\mathbf{a}}^0 = \mathbf{F}^0 - K\mathbf{u}^0 - A(\mathbf{u}^0)\mathbf{u}^0 \tag{4.23}$$

   where $\tilde{\mathbf{a}}^0$ is the instantaneous acceleration neglecting the pressure gradient.

2. Solve the global PPE for the current pressure field.

$$[C^T M_L^{-1} C]p^0 = C^T \tilde{\mathbf{a}}^0 - \dot{g}^0 \tag{4.24}$$

   In LS-DYNA, the prescribed initial conditions and boundary conditions are tested and, if necessary, a projection to a div-free subspace is performed on the initial velocity field, $\mathbf{u}^0$. This guarantees that the flow problem is well-posed, even if the user prescribed initial conditions violate the conditions of Eq. (2.25) – (2.26).

   In practice, the criterion for performing a div-free projection is based upon the RMS divergence error

$$\sqrt{\frac{(C^T\mathbf{u})\cdot(C^T\mathbf{u})}{Nel}} \leq \varepsilon \tag{4.25}$$

   where Nel is the number of elements and ε is a user-specified tolerance typically 10^{-10} to 10^{-7}. If the RMS divergence error is greater than the specified tolerance for the initial candidate velocity field, $\tilde{\mathbf{u}}^0$, then the PPE problem in Eq. (4.26) is solved for λ, and a mass-consistent projection performed using Eq. (4.27).

$$[C^T M_L^{-1} C]\lambda = C^T \tilde{\mathbf{u}}^0 - g^0 \tag{4.26}$$

$$\mathbf{u}^0 = \tilde{\mathbf{u}}^0 - M_L^{-1} C\lambda \tag{4.27}$$

The semi-implicit projection algorithm must respect a relaxed convective stability limit where $CFL \leq O(5-10)$ as described in Gresho and Chan [4]. Because of the semi-implicit treatment of viscous terms, there is no diffusive stability constraint. Computational experiments have demonstrated that $CFL \leq 5$ is a reasonable tradeoff between accuracy and computational cost.

In an unstructured grid, with variable element size, the calculation of the grid *Re* (Reynolds) and *CFL* (Courant-Freidrichs-Levy) numbers uses the element-local coordinates and centroid velocities as described in Chapter 3.

The second-order semi-implicit projection method uses one additional modification to the finite element formulation that derives from the explicit treatment of the advective terms. For advection dominated flows, it is well known that the use of a backward-Euler treatment of the advective terms introduces excessive diffusion. Similarly, Gresho, et al.[1] have shown that forward-Euler treatment of the advective terms results in negative diffusivity, or an under-diffusive scheme. In order to remedy this problem, balancing-tensor diffusivity (BTD), derived from a Taylor series analysis to exactly balance the diffusivity deficit, is adopted. In the one-point quadrature element, the BTD term is simply added to the kinematic viscosity in Eq. (4.28) to form the tensorial diffusivity.

$$\hat{\mu}_{ij} = \mu_{ij} + \rho \frac{\Delta t}{2} u_i u_j \tag{4.28}$$

4.2 Fully-Implicit Projection Method

The fully-implicit projection method relies upon the implicit treatment of the advection terms as well as the viscous terms in the momentum, energy and species transport equations. The fully-implicit method is implemented with automatic time-step selection based on local time truncation error requiring a slightly different start-up procedure than the semi-implicit projection method. The details of the automatic time-step selection may be found in Gresho, et al. [7].

The start-up procedure for the fully-implicit algorithm follows the start-up algorithm outlined in §4.1 to obtain $\mathbf{u}^0$ and p^0, but requires the consistent calculation of the acceleration field at $t = 0$.

4.2.1 Start-Up

Given an initial velocity field $\mathbf{u}^0$, the start-up procedure for the fully-implicit algorithm proceeds as follows.

1. Calculate the partial acceleration, i.e., acceleration neglecting the pressure gradient by solving the mass-matrix problem.
$$M\tilde{\mathbf{a}}^0 = \mathbf{F}^0 - K\mathbf{u}^0 - A(\mathbf{u})\mathbf{u}^0 \tag{4.29}$$
where $\tilde{\mathbf{a}}^0$ is the instantaneous acceleration neglecting the pressure gradient.

2. Solve the global PPE for the current pressure field.
$$[C^T M_L^{-1} C]p^0 = C^T \tilde{\mathbf{a}}^0 - \dot{g}^0 \tag{4.30}$$

3. Calculate the total acceleration at $t = 0$.
$$\dot{\mathbf{u}}^0 = \tilde{\mathbf{a}}^0 - CM_L^{-1} p^0. \tag{4.31}$$

4.2.2 First Time-Step

The first time-step at $t = 0$ proceeds as follows.

1. Compute a predicted velocity at $t = \Delta t$ using the known initial velocity and acceleration,

$$\hat{\mathbf{u}}^1 = \mathbf{u}^0 + \Delta t^0 \dot{\mathbf{u}}^0. \tag{4.32}$$

2. Solve the momentum equations for $\tilde{\mathbf{u}}^1$,

$$\begin{aligned}[M + \Delta t(\theta_K K + \theta_A A(\mathbf{u}^*))]\tilde{\mathbf{u}}^1 = \\ [M - \Delta t[(1-\theta_K)K + (1-\theta_A)A(\mathbf{u}^*)]]\mathbf{u}^0 + \\ \Delta t\{\theta_F \mathbf{F}^1 + (1-\theta)\mathbf{F}^0 + -MM_L^{-1}Cp^0\}\end{aligned} \tag{4.33}$$

where $0 \leq \theta_K, \theta_F, \theta_A \leq 1$ with $\theta_A = \theta_K = \theta_F = 1/2$ for second-order accuracy, and

$$\mathbf{u}^* = \frac{\hat{\mathbf{u}}^1 + \mathbf{u}^0}{2}. \tag{4.34}$$

Remark

Although the fully-discrete momentum equations are presented using a generalized trapezoidal rule in Eq. (4.33) and below in Eq. (4.39), the second-order Adams-Bashforth predictor requires that $\theta_A = \theta_K = \theta_F = 1/2$ be used. This is the default for the fully-implicit projection method. For additional information, see the *CONTROL_CFD_MOMENTUM and *CONTROL_CFD_TRANSPORT keywords in Chapter 9.

3. Solve Eq. (4.19) for the Lagrange multiplier λ and project the approximate velocity to obtain

$$\mathbf{u}^1 = \tilde{\mathbf{u}}^1 - M_L^{-1}C\lambda \tag{4.35}$$

After the velocity update, an updated pressure at time level $n+1$ is obtained via

$$p^{n+1} = p^n + \frac{\lambda}{\Delta t}. \tag{4.36}$$

4. Update the acceleration as

$$\dot{\mathbf{u}}^1 = \frac{2(\mathbf{u}^1 - \mathbf{u}^0)}{\Delta t^0} - \dot{\mathbf{u}}^0. \tag{4.37}$$

4.2.3 General Time-Step

Given a div-free velocity $\mathbf{u}^n$ and pressure p^n with the acceleration vectors $\dot{\mathbf{u}}^n$ and $\dot{\mathbf{u}}^{n-1}$, the fully-implicit algorithm proceeds as follows.

1. Compute a predicted velocity at t^{n+1} using the known velocity and acceleration.

$$\hat{\mathbf{u}}^{n+1} = \mathbf{u}^n + \frac{\Delta t^n}{2}\left[(2+\frac{\Delta t^n}{\Delta t^{n-1}})\dot{\mathbf{u}}^n - \frac{\Delta t^n}{\Delta t^{n-1}}\dot{\mathbf{u}}^{n-1}\right] \tag{4.38}$$

2. Solve the momentum equations treating the simplified form of the linearized equations and perform the projection step, i.e.,

$$\begin{aligned}[M+\Delta t(\theta_K K+\theta_A A(\mathbf{u}^*))]\tilde{\mathbf{u}}^{n+1} = \\ [M-\Delta t[(1-\theta_K)K+(1-\theta_A)A(\mathbf{u}^*)]]\mathbf{u}^n + \\ \Delta t\{\theta_F\mathbf{F}^{n+1}+(1-\theta_F)\mathbf{F}^n + -MM_L^{-1}CP^n\}\end{aligned} \tag{4.39}$$

where

$$\mathbf{u}^* = \frac{\hat{\mathbf{u}}^{n+1}+\mathbf{u}^n}{2}. \tag{4.40}$$

3. Solve for the Lagrange multiplier,

$$[C^T M_L^{-1} C]\lambda = C^T\tilde{\mathbf{u}}^{n+1} - g^{n+1}. \tag{4.41}$$

4. Perform the velocity update.

$$\mathbf{u}^{n+1} = \tilde{\mathbf{u}}^{n+1} - M_L^{-1}C\lambda \tag{4.42}$$

After the velocity update, an updated pressure at time level $n+1$ is obtained via

$$p^{n+1} = p^n + \frac{\lambda}{\Delta t}. \tag{4.43}$$

5. Update the acceleration as

$$\dot{\mathbf{u}}^{n+1} = \frac{2(\mathbf{u}^{n+1}-\mathbf{u}^n)}{\Delta t^n} - \dot{\mathbf{u}}^n. \tag{4.44}$$

6. Update the time step based on the user-specified local time truncation error ε according to

$$\Delta t^{n+1} = \Delta t^n\left\{\frac{\varepsilon}{\|d(\mathbf{u}^{n+1}\|)}\right\}^{\frac{1}{3}}, \tag{4.45}$$

where

$$\|d(\mathbf{u}^{n+1}\|^2 = \frac{1}{\sum_{k=1}^{Ndof} Nnp_k}\left[\sum_{k=1}^{Ndof}\frac{1}{max(u_k^{n+1})^2}\sum_{i=1}^{Nnp_k} d_k(u_k^{n+1})^2\right], \tag{4.46}$$

$Ndof$ is the number of degrees-of-freedom (DOF) per node, e.g., $u,v,w,T,Z1$, and Nnp_k is the number of unknowns per DOF. Here, d_k is the error between the corrected and predicted values of the field variables

$$d(u_k^{n+1}) = \frac{u_k^{n+1}-\hat{u}_k^{n+1}}{3\left(1+\frac{\Delta t^{n-1}}{\Delta t^n}\right)}. \tag{4.47}$$

In LS-DYNA, the time-step prediction is limited by a maximum time-step scale factor

$$\left\{\frac{\varepsilon}{\|d(\mathbf{u}^{n+1}\|)}\right\}^{\frac{1}{3}} \le DTSF, \tag{4.48}$$

where $1.5 \le DTSF \le 2.0$. In addition, a user-specified upper limit, Δt_{max}, is placed on the time-step size.

Chapter 5

Boundary Conditions and Source Terms

This chapter summarizes the boundary conditions for the incompressible flow solver in LS-DYNA. Keywords that are relevant to the boundary conditions are summarized in Chapter 9 and fully documented in the LS-DYNA Keyword User's Manual [27].

5.1 Node and Segment Sets

Sets provide a generalized concept for grouping nodes, elements, and materials. In LS-DYNA, sets provide the basic entity for the prescription of boundary conditions and are used for all keyword input. Node sets consist of an arbitrary list of nodes that are treated as one entity for the application of nodal boundary conditions. Segment sets consist of an arbitrary list of quadrilateral edges in two dimensions and a list of hexahedral faces in three dimensions. Figure 5.1 shows a node set associated with inflow conditions and a segment set associated with outflow boundary conditions for an external flow problem. In addition to node and segment sets, the incompressible flow solver uses shell and solid sets to identify elements with specific attributes. Additional information on sets and the *SET keywords may be found in the LS-DYNA keyword manual [27].

5.2 Nodal Boundary Conditions

The prescription of nodal boundary conditions for the incompressible flow solver encompasses all nodal degrees-of-freedom, e.g., velocity, temperature, mass concentration species, and turbulent kinetic energy. These boundary conditions are typically referred to as essential or Dirichlet boundary conditions and fix the nodal values of the field variable according to a prescribed value or function of time. All nodal boundary conditions for the incompressible flow solver are prescribed using the *BOUNDARY_PRESCRIBED_CFD_NODE and *BOUNDARY_PRESCRIBED_CFD_SET keywords.

As an example of nodal boundary conditions, consider the prescription of boundary conditions based on the flow domain shown in Figure 5.1. Inflow boundary velocity conditions that emulate free-stream conditions are prescribed as $u_1 = U$ and $u_2 = 0$ on Γ_i. No-slip and no-penetration velocity boundary conditions are prescribed at the cylinder wall, $u_i = 0$ on Γ_s.

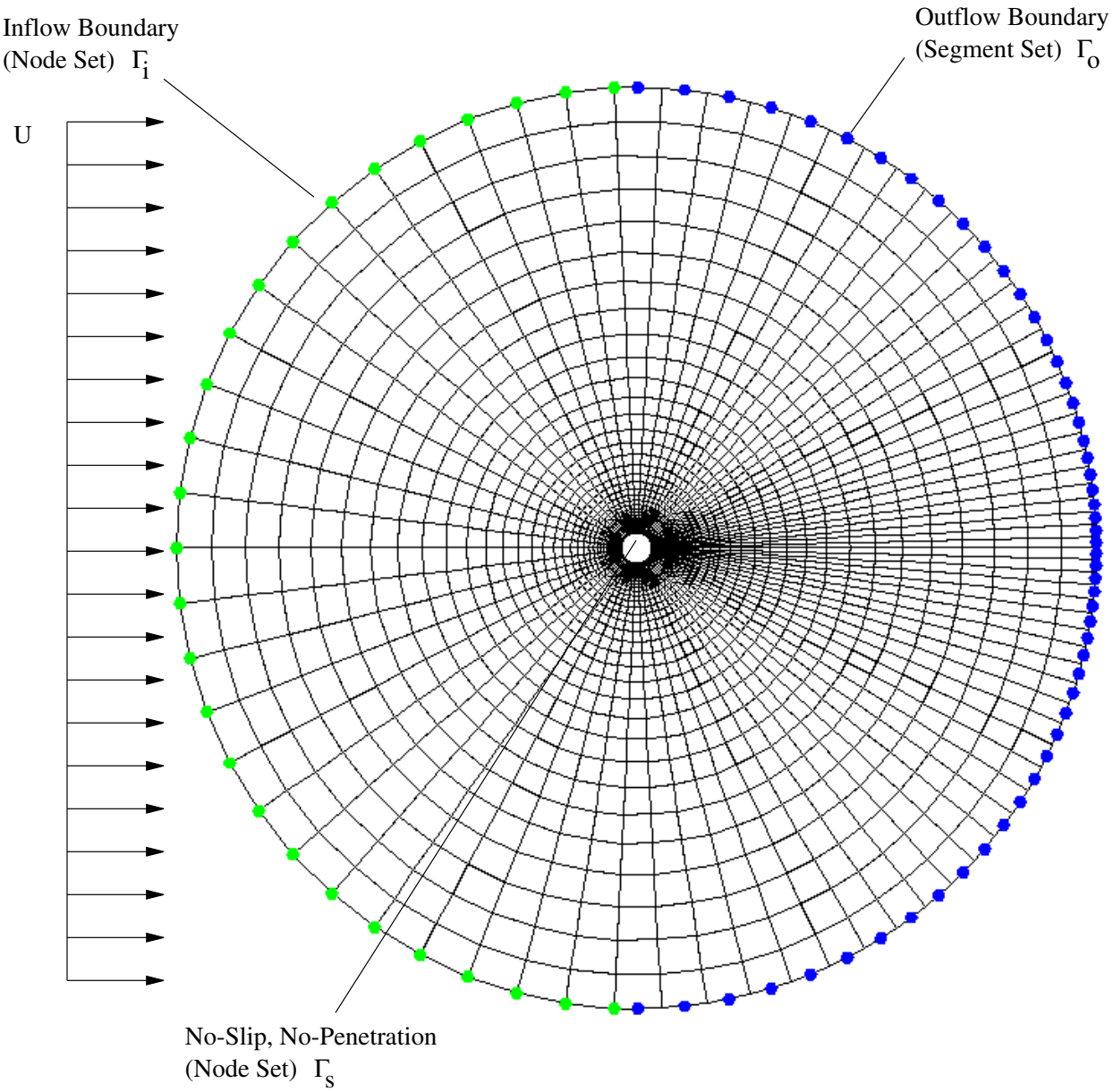


Figure 5.1: Flow domain with inflow Γ_i, outflow Γ_o, and no-slip, no-penetration Γ_s boundaries.

The boundary conditions for the energy equation, Eq. (2.13), consist of a prescribed temperature or heat flux rate. For example, a prescribed temperature at the cylinder wall is $T(x_i,t) = \hat{T}$ on Γ_s. Turning attention to the species transport equations, boundary conditions for Eq. (2.9) may consist of either a prescribed concentration or a mass flux rate. In the case of a binary mixture with a prescribed free-stream concentration, $Z_1(x_i,t) = \hat{Z}_1$ on Γ_i where $\hat{Z}_1$ is the known value of concentration for species 1.

5.3 Traction Boundary Conditions

The formulation of the weak-form of the momentum equations Eq. (2.1) yields traction boundary conditions also known as natural boundary conditions. In terms of the shape functions N_a, the traction boundary conditions may be computed as

$$F^e_{i_a} = \int_\Gamma^e N_a \, \hat{f}_i(x_i,t) \, d\Gamma, \tag{5.1}$$

where $\hat{f}_i$ are the components of the prescribed traction. Here, $1 \leq i \leq Ndim$ and $1 \leq a \leq Nnpe$ where $Ndim$ is the number of space dimensions and $Nnpe$ is the number of nodes per element.

Alternatively, the boundary conditions may be written in terms of the stress,

$$F^e_{i_a} = \int_\Gamma^e N_a \, \hat{\sigma}_{ij} n_j \, d\Gamma, \tag{5.2}$$

where $\hat{\sigma}_{ij}$ is the prescribed stress and n_j is the outward normal for the domain boundary.

In terms of the pressure and strain-rate, the traction boundary conditions are

$$F^e_{i_a} = \int_\Gamma^e N_a \, \left\{ -\hat{p}\delta_{ij} + 2\mu \, \hat{\varepsilon}_{ij} \right\} n_j. \tag{5.3}$$

By default, homogenous traction boundary conditions are applied unless other boundary conditions are prescribed, i.e., homogeneous traction (natural) boundary conditions are the *do-nothing* boundary conditions. The traction and velocity boundary conditions can be mixed. In a two-dimensional sense, mixed boundary conditions can consist of a prescribed normal traction and a tangential velocity. For example, at the outflow boundary in Figure 2.1, a homogeneous normal traction and vertical velocity on Γ_2 constitutes a valid set of mixed boundary conditions. A detailed discussion of boundary conditions for the incompressible Navier-Stokes equations may be found in Gresho and Sani [5]. Note that this boundary condition provides an important component for the coupling between fluid and structural components for coupled problems and is computed internally for this class of problems.

5.3.1 Pressure Boundary Conditions

In many practical situations, the viscous contributions in Eq. (5.3) may be neglected. This is typically the case in situations where the viscosity is small, i.e., the Reynolds number is relatively large. In this case, the viscous terms are ignored leaving only the pressure contribution to the traction boundary conditions,

$$F^e_{i_a} = \int_\Gamma^e N_a \, \{-\hat{p}\} \, n_i, \tag{5.4}$$

where $\hat{p}$ is the prescribed pressure.

Due to the common occurrence of this type of limiting traction boundary condition, LS-DYNA refers to this as a *pressure boundary condition* which reflects its use in finite element models. The *BOUNDARY_PRESSURE_CFD_SET keyword is used to prescribe the pressure as a function of time, $\hat{p}(t)$ on a boundary segment set.

5.4 Outflow Boundary Conditions

In many problems, it is necessary to truncate the physical domain resulting in an artificial outflow boundary. In the presence of strong vortical structures the homogeneous natural (do-nothing) boundary conditions can result in a global pressure response that is physically unrealistic. Heinrich, et al. [53] present a summary of this problem and suggest several remedies.

In LS-DYNA, *outflow* boundary conditions that similar to the pressure boundary condition in Eq. (5.4) are used. However, the outflow boundary conditions use a known pressure distribution that has been computed during the solution procedure. This boundary condition results in an equilibrating force applied on the outflow boundary,

$$F_{i_a}^e = \int_\Gamma^e N_a \ \{-p^n\}\, n_i, \tag{5.5}$$

where p^n is the pressure field computed at time t^n.

The outflow boundary condition is activated using the *BOUNDARY_OUTFLOW_CFD_SET keyword which applies the outflow boundary conditions on a segment set basis.

5.5 Body Forces

The presence of body forces in the momentum equations results in element-level force contributions. At the element-level, the forces are

$$F_{i_a}^e = \int_\Omega N_a \ \rho f_i \ d\Omega, \tag{5.6}$$

where f_i represents the body-force per unit volume.

In thermal-convection problems where a Boussinesq fluid is appropriate, the body-force due to buoyancy is

$$F_{i_a}^e = \int_\Omega N_a \ \rho g_i \ \beta(T - T_{ref}) \ d\Omega. \tag{5.7}$$

where, g_i is the acceleration due to gravity, β is the coefficient of thermal expansion, and T_{ref} is a reference temperature. The specification of the gravity vector, reference temperature and fluid properties is achieved with the *MAT_CFD_*OPTION* keyword.

Remark

By default, the fluid temperature is initialized to the reference temperature T_{ref} specified in the *MAT_CFD_*OPTION* keyword. Each fluid is initialized material-by-material resulting in *built-in* temperature initial conditions.

5.6 Flux Boundary Conditions

The heat flux rate at a boundary is prescribed as

$$Q_a^e = \int_\Gamma N_a \, \hat{q}_i(x_i,t) \, n_i \, d\Gamma \tag{5.8}$$

where $\hat{q}_i$ is the known flux rate through the boundary with normal n_i. Using the constitutive relationship between temperature and heat-flux is

$$\hat{q}_i(x_i,t) = -k_{ij}\frac{\partial T}{\partial x_j}, \tag{5.9}$$

where k_{ij} is the thermal conductivity tensor. The homogeneous form of the heat-flux boundary condition is the default and represents a perfectly insulated, i.e., zero heat-flux boundary.

The heat flux rate may also be prescribed in terms of a convective heat transfer coefficient,

$$\hat{q}_i(x_i,t) \, n_i = h(T - T_\infty), \tag{5.10}$$

where h is the heat transfer coefficient, and T_∞ is a reference temperature.

For the species transport equations, the prescribed mass flux rate for species-1 is

$$\dot{m}_a^e = \int_\Gamma N_a \, \hat{J}_{1_i} \, n_i \, d\Gamma, \tag{5.11}$$

where $\hat{J}_{1_i}$ is the known flux rate through the boundary with normal n_i. (For simplicity in the presentation, only the boundary conditions for species-1 are presented here.) Using the constitutive relation between species concentration and mass-flux rate,

$$\hat{J}_1(x_i,t) = -\rho D_{1_{ij}}\frac{\partial Z_1}{\partial x_j}, \tag{5.12}$$

where $D_{1_{ij}}$ is the tensorial mass diffusivity. The homogeneous form of the mass-flux boundary condition is the default and represents a boundary where the gradient of the species in the boundary normal direction is zero.

The prescribed flux rate may also be specified in terms of a convective mass transfer coefficient as

$$\hat{J}_{1_i} \, n_i = h_{D_\infty}(Z_1 - Z_{1_\infty}), \tag{5.13}$$

where h_{D_∞} is the mass transfer coefficient and Z_{1_∞} is a reference species concentration.

5.7 Pressure Levels

In incompressible flow, the pressure is typically only known up to an additive constant – the hydrostatic pressure level. The prescription of a specific hydrostatic pressure level is achieved through the *CONTROL_CFD_PRESSURE keyword. For the $Q1Q0$ element technology, setting the hydrostatic pressure level requires a shell-set in 2-D or a solid-set in 3-D to identify those elements for which the pressure level will be fixed.

Chapter 6

Pressure Solution Methods

This chapter presents a survey of solution strategies, stabilization techniques and linear algebra procedures used in LS-DYNA for calculating the pressure in the time-dependent Navier-Stokes equations. The primary focus is on the treatment of the pressure-Poisson equation deriving from the index-1 formulations of the Navier-Stokes equations presented in Chapters 3 – 4. Operational experience has shown that, depending on the solution strategy, the pressure solution can consume $80-90\%$ of the CPU time per time step in semi-implicit and fully-implicit projection methods – even more when explicit time integration is used. Based on extensive operational experience with a variety of solution strategies, the combination of a stabilized pressure-Poisson operator with an A-conjugate projection and SSOR preconditioned conjugate gradient method has been found to yield the overall best performance relative to the re-solve cost of a high performance direct solver.

6.1 The Saddle Point Problems

The pressure-Poisson equation (PPE) in the projection algorithms presented in Chapter 4 arises due to the use of a Helmholtz velocity decomposition that decomposes the velocity into div-free and curl-free components. In a finite element context, the Helmholtz velocity decomposition with the concomitant div-free constraint, $C^T\mathbf{u}^{n+1} = g^{n+1}$, is written as

$$\begin{bmatrix} M_L & C \\ C^T & 0 \end{bmatrix} \begin{Bmatrix} \mathbf{u}^{n+1} \\ \lambda \end{Bmatrix} = \begin{Bmatrix} M_L\tilde{\mathbf{u}}^{n+1} \\ g^{n+1} \end{Bmatrix}, \tag{6.1}$$

where, in general, $C^T\tilde{\mathbf{u}}^{n+1} \neq g^{n+1}$.

The Schur complement of the *projection* operator in Eq. (6.1) yields the PPE in Eq. (4.19). In the ensuing discussion, the term *projection* operator is used loosely to describe Eq. (6.1). However, the *projection* operator is not to be confused with the real discrete projection operators, $P^h = I - M_L^{-1}C[C^TM_L^{-1}C]^{-1}C^T$ and $Q^h = I - P^h$ (see Gresho and Chan [4]).

In the explicit time integration algorithm presented in Chapter 3, a saddle-point problem analogous to Eq. (6.1) may also be formulated in terms of the pressure p^n, partial acceleration $\tilde{\mathbf{a}}^n$, and acceleration $\dot{\mathbf{u}}^n$ as.

$$\begin{bmatrix} M_L & C \\ C^T & 0 \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{u}}^n \\ p^n \end{Bmatrix} = \begin{Bmatrix} M_L\tilde{\mathbf{a}}^n \\ \dot{g}^n \end{Bmatrix}. \tag{6.2}$$

In this case, the acceleration and pressure are used rather than velocity and pressure increment highlighting one of the primary differences between the projection and the explicit methods. The explicit method segregates acceleration and pressure, which can be done legitimately, while the projection methods attempt to decouple velocity and pressure. Similar to the projection methods, the pressure-Poisson problem in Eq. (3.6) is simply the Schur complement of Eq. (6.2). For both algorithms, the PPE and the saddle-point operators are symmetric and in general singular due in part the fact that the pressure is known only to within a constant representing the hydrostatic pressure.

6.2 $Q1Q0$ Stabilization

Although the $Q1Q0$ element has been condemned by theoreticians for its weakly-singular modes, this element has long been the workhorse for incompressible flow and continues to be widely used. The unstable modes of the $Q1Q0$ element for incompressible flow have been investigated by Sani, et al. [54, 55] and more recently by Griffiths and Silvester [56]. Griffiths and Silvester demonstrate that for problems of physical interest, the $Q1Q0$ element will converge to the true solution in the limit as $h \to 0$. A new convergence proof for the $Q1Q0$ element may be found in Gresho and Sani [5].

The ensuing discussion summarizes the pressure stabilization techniques in LS-DYNA that are compatible with the $Q1Q0$ element and that circumvent the usual Babuška-Brezzi div-stability condition. In LS-DYNA, there are two pressure stabilization techniques, one that is referred to as "global jump stabilization", and one that is referred to as "local jump stabilization". The global jump stabilization (first proposed by Hughes and Franca [57]) and the local jump stabilization techniques of Silvester and his coworkers[58, 59, 60, 61] are applied to the PPE problem for both the explicit and projection methods. In effect, the jump stabilization techniques provide an a priori filter for the weakly unstable pressure modes associated with the $Q1Q0$ element.

The stabilized $Q1Q0$ element yields a regularized saddle-point problem for the projection method,

$$\begin{bmatrix} M_L & C \\ C^T & -S \end{bmatrix} \begin{Bmatrix} \mathbf{u}^{n+1} \\ \lambda \end{Bmatrix} = \begin{Bmatrix} M_L \tilde{\mathbf{u}}^{n+1} \\ g^{n+1} \end{Bmatrix}, \tag{6.3}$$

where S is a symmetric positive semi-definite matrix. Here, pressure stabilization results in an approximate projection method since the associated stabilized PPE is no longer constructed using only the discrete div and grad operators. The Schur complement of Eq. (6.3), i.e., the stabilized pressure Poisson equation (PPE), is

$$\left[C^T M_L^{-1} C + S\right] \lambda = C^T \tilde{\mathbf{u}}^{n+1} - g^{n+1} \tag{6.4}$$

where S remains to be defined for the global and local jump stabilization techniques.

The global jump stabilization formulation attempts to control the jump in pressure across element boundaries, and results in a PPE that is perturbed by a pressure-diffusion operator. The off-diagonal entries in the global jump stabilization matrix are defined as

$$S_{IJ} = \beta \frac{|[C^T M_L^{-1} C]_{IJ}|}{\Gamma_{IJ}} \int_{\Gamma_{IJ}} \|\psi_I\| \|\psi_J\| d\Gamma, \tag{6.5}$$

where I and J identify adjacent elements that share a common face as shown in Figure 6.1a. Here, Γ_{IJ} represents the shared inter-element boundary, $\|\cdot\|$ is the jump operator, β is a non-dimensional parameter, and ψ_I is the pressure basis function for element I. In two dimensions, Γ_{IJ} represents the length of the

element edge shared by element I and J, and in three dimensions, it represents the area of the shared face. The inclusion of the PPE term, $|[C^T M_L^{-1} C]_{IJ}|$, in Eq. (6.5) yields proper dimensionality of the stabilization matrix, accounts for scaling due to irregular elements, and still preserves the symmetry of the original PPE. For alternative scaling procedures, see Chan and Sugiyama [62]. The diagonal contributions are calculated as

$$S_{II} = \beta \sum_J \frac{|[C^T M_L^{-1} C]_{IJ}|}{\Gamma_{IJ}} \int_{\Gamma_{IJ}} \|\psi_I\| \|\psi_J\| d\Gamma, \tag{6.6}$$

where the sum on J indicates a summation over all shared faces of element I. This formulation preserves the row-sum-zero property of the original PPE. The global jump stabilization is simple to implement from a linear algebra point of view since the contributions to S are simply augmented row-entries from the original PPE.

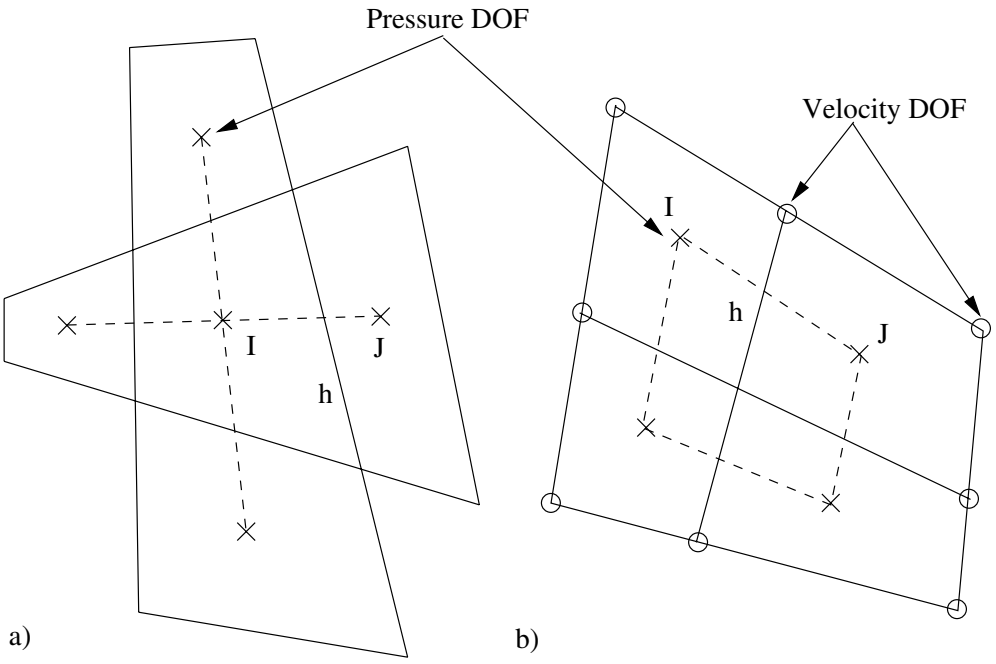


Figure 6.1: Element configuration for pressure stabilization: a) global jump, b) local jump.

In contrast to the global method, the local stabilization procedure relies on the construction of macro-elements that contain one velocity node per edge of the macro element in two dimensions and one velocity node per face in three dimensions. For the local stabilization, the entries of the stabilization matrix are calculated according to Eq. (6.5) and (6.6) but only for the faces shared with elements in the same macro-element as shown in Figure 6.1b.

Since the same scaling is used for both the local and global jump stabilization, the only remaining parameter to be determined is β. From the inherited scaling of the stabilization matrix, $\beta = 0$ provides the limit where no-stabilization is applied. For $\beta = 1$, the stabilization matrix will have entries of the same order as the original PPE. In the context of Stokes flow, Norburn and Silvester [61] bounded β by computing the extremal eigenvalues for the stabilized Schur complement as a function of β. They found that $0.01 \leq \beta \leq 0.1$ minimizes the distance between the extremal eigenvalues. Numerical testing has shown that in general, values of β in this range yield acceptable results for the stabilized PPE as well.

Figure 6.2 shows the variation of the iteration count for SSOR preconditioned conjugate gradient as a function of β for a 2-D vortex shedding problem and 3-D flow past a circular cylinder using both local and global jump stabilization. This plot shows that for $\beta \geq 0.002$ in 2-D ($\beta \geq 0.004$ in 3-D), there is very

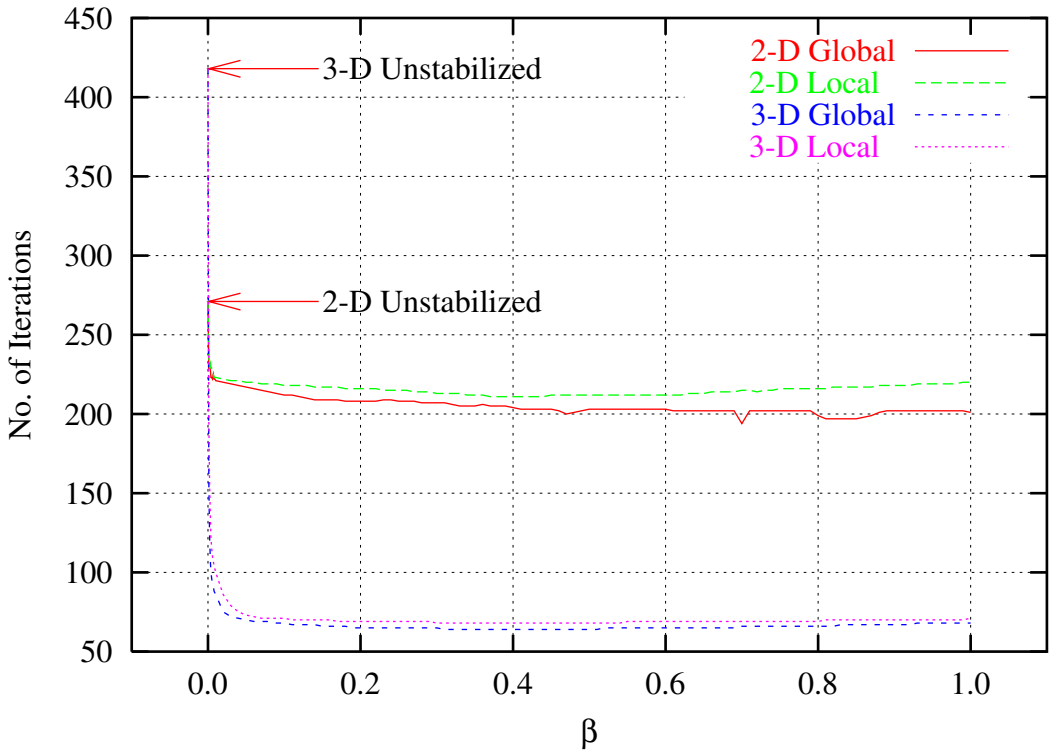


Figure 6.2: Variation of iteration count vs. stabilization parameter for a 2-D $Re = 100$ vortex shedding mesh ($Nel = 11200$) and 3-D $Re = 100$ flow past a cylinder ($Nel = 22380$). (The iteration count for the un-stabilized PPE is indicated by an arrow for the 2-D and 3-D problems.)

little variation in the iteration count with respect to β. In addition, the benefit of stabilization appears to be considerably larger for 3-D, a factor of 6 reduction in iteration count for 3-D compared to a factor of about 1.3 for the 2-D case. The marked decrease in iteration count in 3-D is undoubtedly a consequence of the relatively larger null space for the PPE in three-dimensions. Operational experience has shown that $0.01 \leq \beta \leq 0.1$ works well for most applications while $\beta \geq 0.25$ can often yield inaccurate velocity fields.

The effect of pressure stabilization on the convergence rate is shown in Figure 6.3 which shows convergence histories for coarse, medium and fine grids for a vortex shedding problem. Here, a convergence criteria of $\varepsilon \leq 10^{-12}$ was used to obtain an initial RMS divergence of 10^{-8} using global stabilization. Similar results may be found in Gresho and Sani [5] where the local and global jump stabilization formulations have been applied to a 3-D problem using an ICCG(0) preconditioner with similar improvements in convergence rate.

6.3 The Projection CG Method

The solution of the time-dependent incompressible Navier-Stokes requires the repeated solution of the PPE problem where the coefficient matrix is fixed and the right-hand side changes each time step. To address this aspect of solving the PPE, an A-conjugate projection is integrated with the iterative solution of the PPE in order to use solution information from the previous time steps. In the ensuing discussion, the PPE problem is cast as $Ax = b$ where $A = C^T M_L^{-1} C$ or $A = C^T M_L^{-1} C + S$, $x = \lambda$ and $b = C^T \tilde{\mathbf{u}} - g^{n+1}$.

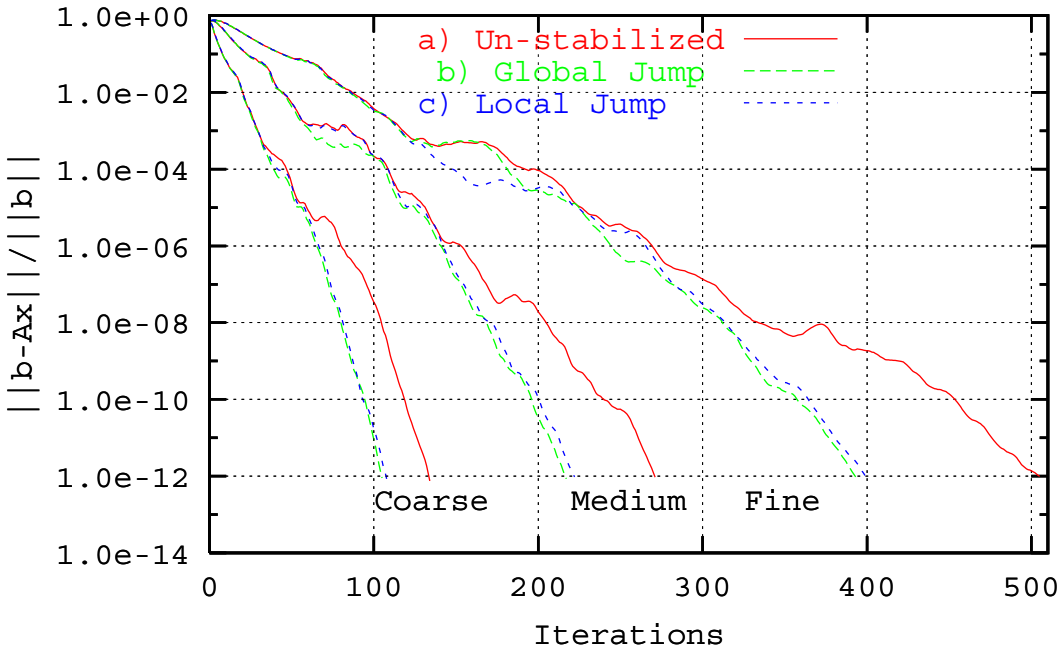


Figure 6.3: Convergence histories for SSOR preconditioned CG applied to the un-stabilized PPE, b) the PPE with the global jump stabilization, and c) the PPE with the local jump stabilization for the coarse (2800 elements), medium (11200 elements) fine resolution (44800 elements) vortex shedding mesh (see Christon [63]).

The use of an A-conjugate projection as a pre-processing step for the solution of the linear system, $Ax = b$, follows the development presented by Fischer [64] with extensions that permit seeding the A-conjugate vectors. Related work on solving linear systems with multiple right-hand sides may be found in Saad [65] and Chan and Wan [66].

To begin the development, the idea of a pre-processing A-conjugate projection step relies on minimizing the distance between the solution at a given time step, x^n, and the base vectors Φ in the A-norm. Here, Φ is a set of A-conjugate vectors derived from N prior solutions to $Ax = b$ where $\Phi = \{\phi_i, i = 1, N\}$. As will be demonstrated in §6.4, $N = 5$ to $N = 10$ is a reasonable choice.

Let the initial "guess" for a solution at time-step n be given by

$$\overline{x} = \sum_{i=1}^{N} \alpha_i \phi_i. \tag{6.7}$$

Let the difference between the solution at time-step n and the initial guess, $\overline{x}$, be defined as

$$\|e\|_A = \|x^n - \overline{x}\|_A, \tag{6.8}$$

where $\|\cdot\|_A = \sqrt{(\cdot)^T A(\cdot)}$.

With this definition,

$$\|x^n - \overline{x}\|_A^2 = (x^n)^T A x^n - \overline{x}^T A x^n - (x^n)^T A\overline{x} + (\overline{x})^T A\overline{x}, \tag{6.9}$$

and for A being symmetric positive definite,

$$\|x^n - \overline{x}\|_A^2 = (x^n)^T A x^n - 2(x^n)^T A\overline{x} + (\overline{x})^T A\overline{x}. \tag{6.10}$$

Now, substituting Eq. (6.7) and using the fact that Φ is a set of A-conjugate vectors,

$$||e||_A^2 = (x^n)^T A x^n - 2\sum_{i=1}^{N} \alpha_i (\phi_i)^T A x^n + \sum_{i=1}^{N} \alpha_i^2. \tag{6.11}$$

Minimization of $||e||_A$ with respect to α_i yields

$$\alpha_i = (\phi_i)^T A x^n = (\phi_i)^T b^n. \tag{6.12}$$

Thus, given a set of A-conjugate vectors, Φ, the best approximation to x^n that minimizes the error in the A-norm is obtained by projecting the right-hand-side, b^n, at time-step n onto the set of base vectors, Φ. This suggests the following solution procedure.

Algorithm 1 *A-conjugate projection CG method.*

1. $\alpha_i = (\phi_i)^T b^n$.
2. $\overline{x} = \sum_{i=1}^{N} \alpha_i \phi_i$ *where N is the number of A-conjugate base vectors.*
3. *Solve* $A\Delta x^n = r^n$ *using the conjugate gradient method where* $r^n = b^n - A\overline{x}$.
4. *Update the solution,* $x^n = \overline{x} + \Delta x^n$.
5. *Update the base vectors,* Φ *to include new information from the last solution.*

As an aside, the solution at a given time level may be written as

$$x^n = \Delta x^n + \sum_{i=1}^{N} \alpha_i \phi_i. \tag{6.13}$$

Updating Φ

For the initial solution, or when the number of existing base vectors exceeds N, the basis is started by normalizing the solution as

$$\phi_1 = \frac{x^1}{||x^1||_A}. \tag{6.14}$$

For all other cases, a solution vector is a candidate for addition to Φ only when it contains non-trivial information not already present in Φ. Here, the basic idea is that Δx is A-conjugate to $\overline{x}$ as well as to the individual base vectors ϕ_i. Therefore, the addition of a new base vector should be based on criteria that guarantees that new information is being added to the existing base vectors. Addition of a solution vector

(or a part of a solution vector) proceeds by first computing the part of the solution that is not contained in the basis as

$$\psi_{l+1} = x^n - \sum_{i=1}^{l} \alpha_i \phi_i. \tag{6.15}$$

Now, each ϕ_i is A-conjugate to ψ_{l+1} by construction and is added to the basis as

$$\phi_{l+1} = \frac{\psi_{l+1}}{\|\psi_{l+1}\|_A}. \tag{6.16}$$

The idea of seeding the A-conjugate vectors in Φ with several initial vectors has been suggested by T.J.R. Hughes [67] as a mechanism for reducing cost of the initial linear solves, i.e., the first PPE solve when Φ is empty. This may be achieved by selecting several global polynomials and even random vector as seeds for Φ. Construction of the A-conjugate vectors in Φ proceeds according to the algorithm above.

Experimentation with vectors representing a random solution, hydrostatic and simple polynomials have been tested with limited success. In the case of simple flows, e.g., Pouiselle flow, the linear polynomials contribute directly to the initial solution and reduce the number of CG iterates dramatically. However, in moderately complex domains, the simple polynomials provide little reduction in computational cost. It is thought that initial polynomial fields that respect the implied pressure boundary conditions may provide an effective set of seed vectors, but this has not yet been implemented.

To illustrate the use of the A-conjugate projection, Figure 6.4a) shows a snapshot of the pressure field for a $Re = 100$ vortex shedding computation. In addition, Figures 6.4b) – 6.4f) show snapshots of the Φ vectors based on the previous five time steps. It is clear that ϕ_1 provides primarily long wavelength information while the other 4 vectors provide detailed information about the wake. Thus, the vectors $\phi_2 - \phi_5$ may be viewed as short wavelength corrections to ϕ_1 that yield the best approximation to the current pressure field. The A-conjugate projection procedure, in effect, selects the appropriate information from each ϕ vector in order to minimize the residual in the A-norm before performing any CG iterations.

The A-conjugate projection procedure retains both long and short wavelength information, and in this sense, the procedure may be viewed as an approximate means of deflating the eigenvalue spectrum for the PPE. The combination of the A-conjugate projection method, PPE stabilization and SSOR preconditioning has proved to be the most computationally efficient method for solving the PPE.

6.4 PPE Solver Performance

This section summarizes the performance of the PPE solvers via several computational experiments performed using the A-conjugate projection strategy with several variants of the preconditioned conjugate gradient method. In addition, the effect of pressure stabilization on the iterative solver is considered for both local and global stabilization. The CG variants include a diagonally scaled conjugate gradient (JPCG) solver, an SSOR preconditioned CG method, and an SSOR preconditioned CG solver implemented using the Eisenstat transformation – the ESSOR CG solver. Each CG solver used a row-compressed storage scheme for the PPE. The SSOR preconditioners were implemented in a stand-alone form (SSOR-PCG) that requires separate matrix-vector preconditioning functions, and in the ESSOR-PCG form using the transformation first suggested by Eisenstat [68] to reduce the operations count in the preconditioned conjugate gradient method. (See also Ortega [69] and Eisenstat, et al. [70].)

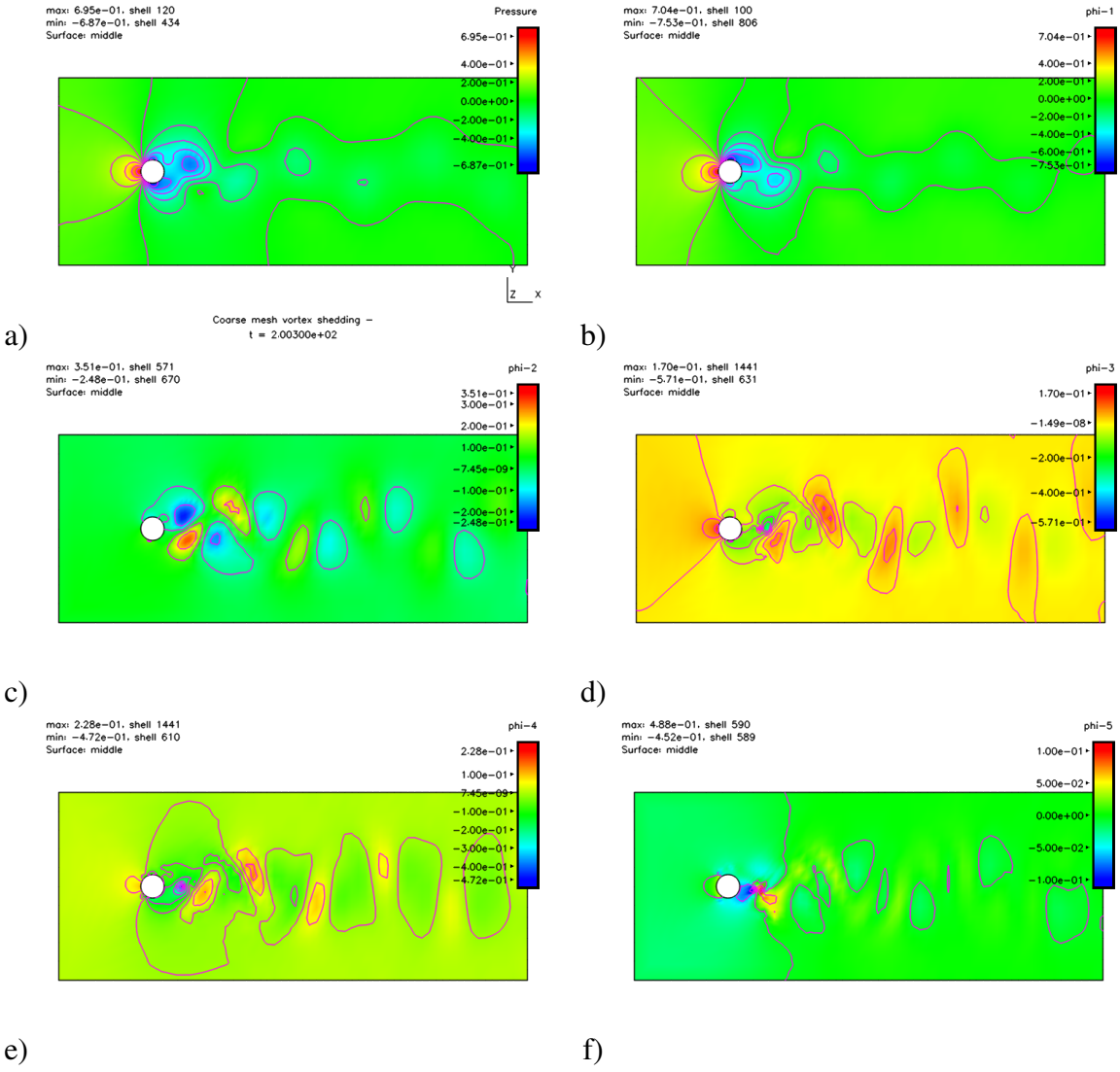
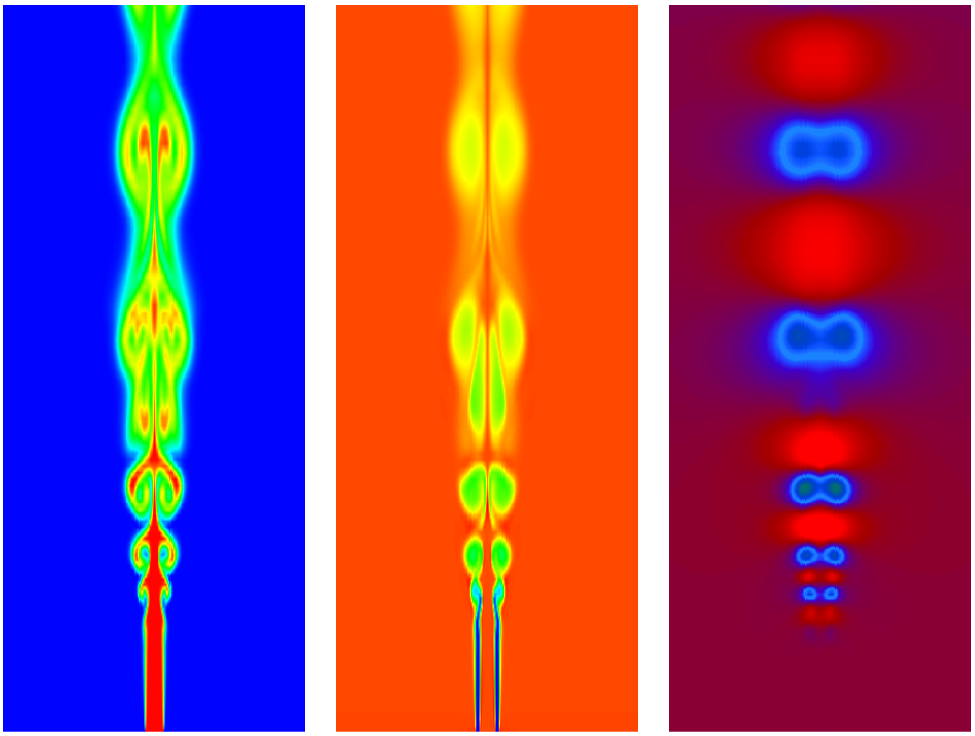


Figure 6.4: a) Snapshot of the pressure field during vortex shedding, and five ϕ fields b) – f) based on pressure solution at the five prior time steps.



a) Temperature b) Vorticity c) Pressure

Figure 6.5: Snapshot of a) temperature field, b) z-vorticity field and c) pressure for the 2-D momentum driven jet, $Re = 3561$, $Fr = 326$. (The temperature, vorticity and pressure fields have been reflected about the vertical centerline for presentation purposes.)

The first series of computations used to evaluate the PPE solution methods were carried out for a momentum driven slot jet with $Re = 3561$ based on a 15 mm slot width and $Fr = 326$ ($Fr = v^2/g\beta\Delta T L_c$ where $g = 9.81\ m/s^2$, $\beta\Delta T = 1/3$, $L_c = 15\ mm$, and $v = 4\ m/s$). In this computation, an energy equation was solved in conjunction with the Navier-Stokes equations using the Boussinesq approximation. The initial conditions consist of a div-free velocity field with a free-field temperature of $300K$, an inlet temperature of $400K$. The working fluid is air with a unit Prandtl number, $Pr = \nu/\alpha$ resulting in $Pe = 4000$ where $Pe = RePr$. Here, the relatively large Froude number indicates that the influence of buoyancy forces is small compared to the inertial forces, i.e., a momentum driven jet.

Figure 6.5 shows snapshots of the temperature, vorticity and pressure fields from the momentum driven jet simulation. In this computation, natural boundary conditions were applied at the far-field boundary removing the hydrostatic mode from the PPE. Table 6.1 and 6.2 show the results of the jet computations in terms of the average iteration count and grind time for 1000 time steps. Here, the grind times are normalized with respect to the grind time for the same computation using the PVS direct solver [71, 72] where a single factorization is performed during initialization and a single re-solve is carried out at each time step. All computations in this comparison were performed using a 500 MHz single processor DEC Alpha.

The solution time due to the PPE re-solve was 15.2% for the semi-implicit projection method and 60.0% for the explicit algorithm using the direct solver. This relatively low percentage is due to the fact that a single, off-line factorization is performed with one re-solve per time step. For the PPE, the ideal situation would be to have a normalized grind-time using the iterative solvers that approaches or beats the re-solve cost for the PVS direct solver. Note that the CPU time associated with the PVS re-solve scales roughly as

$Nel \times Nb$ where Nb is the $1/2$ bandwidth of the PPE.

As Tables 6.1 and 6.2 show, there is a small but noticeable effect of the stabilization on the iteration count and grind-time for the 2-D jet for both the explicit and semi-implicit projection algorithms. This is due to the fact that for all computations, a convergence criteria of $\varepsilon = 10^{-5}$ was used. As shown in Figure 6.3, this level of convergence limits makes the effect of the stabilization appear reduced. However, the SSOR preconditioning reduces the iteration count by nearly a factor of 3 without the A-conjugate projection. The use of the Eisenstat formulation further reduces the grind time by a factor of approximately 1.6.

For the explicit algorithm, the use of 5 base-vectors for the A-conjugate projection algorithm reduces the iteration count by nearly a factor of 3 and the grind time by nearly a factor of 3 regardless of the preconditioner. For the semi-implicit projection algorithm, the effect is reduced somewhat resulting in a factor of about 2.25 reduction in the iteration count and better than a factor of 2 reduction in the grind time.

The combination of the ESSOR-PCG algorithm and 10 projection vectors results in grind times that are within a factor of 3 of the grind-time using the PVS solver. For the projection algorithm, the grind times are within a factor about 2 emphasizing the additional cost of the semi-implicit treatment of the momentum equations. Relative to the basic JPCG solver the ESSOR-PCG solver with 10 projection vectors results in a reduction of nearly 9 for the explicit algorithm and nearly 4 for the projection algorithm. Interestingly, the storage cost of a separate preconditioning matrix is approximately equivalent to 10 projection vectors. Although further reductions in the execution times may be had with increasing number of projection vectors, the increased storage cost and diminishing reduction in the computational costs makes suggests that 5 to 10 vectors is nearly optimal.

The second series of computations were carried out for a three-dimensional channel flow past a circular cylinder with $Re_D = 100$. Figure 6.6 shows snapshots of the computation that was motivated by a suite of benchmark problems used as "round-robin" tests to evaluate incompressible flow solution methods developed under the Deutsche Forschungsgemeinschaft (DFG) Priority Research Program, "Flow Simulation on High Performance Computers". Figure 6.6a shows a snapshot of the 3-D z-vorticity field. Isosurfaces of the corresponding pressure field are shown in Figure 6.6b.

In this computation, a parabolic velocity profile was prescribed at the inlet to the channel with no-slip boundaries on the channel and cylinder walls. Natural boundary conditions were used at the channel outflow. The prescription of $u_i n_i = 0$ on the channel walls results in a two-dimensional checkerboard pressure mode that exists in each plane of three-dimensional elements normal to the flow direction. In this situation, pressure stabilization is necessary to filter the checkerboard modes. For this reason, all computations were performed using the pressure stabilization outlined in §6.2.

Tables 6.3 and 6.4 report the grind times and iteration counts for the channel flow problem using the explicit and semi-implicit projection methods respectively. The results in Table 6.3 show that the cost of the PPE solution dominates the solution time per time step. In contrast, the solution of the stabilized PPE in the projection algorithm is less than one-third of the computational cost per time step when the ESSOR-PCG solver is used. Again, the best overall grind times are found using the ESSOR-PCG solver combined with 5 to 10 base vectors where a speedup of 3 to 4 is observed relative to the basic JPCG solver. The influence of the large half-bandwidth of the PPE is seen on the grind times resulting in normalized grind times approaching unity for the ESSOR-PCG solver.

Based on the results for a range of flow problems, the use of a stabilized consistent pressure-Poisson operator, an A-conjugate projection technique, and SSOR preconditioning with the conjugate-gradient method yields the overall best performance relative to the re-solve cost of a direct solver. The effect of

Table 6.1: Effect of the A-Conjugate projection on the PPE solution time for the 2-D momentum driven jet. The grind times for the explicit algorithm have been normalized with respect to the solution time for 1000 time steps using a direct solver [71, 72]. N_{IT} is the average number of iterations required per time step to solve the PPE problem. ($Nel = 11250$, $Nnp = 11466$, $N_b = 146$, direct solver grind time: 1.2254×10^{-5} CPU seconds per element per time step)

Explicit Algorithm – No Stabilization									
	JPCG			SSOR-PCG			ESSOR-PCG		
No. of Vectors	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE
0	353	26.52	98.9	120	19.89	98.5	109	12.22	97.3
5	114	8.81	96.8	38	6.65	95.7	35	4.20	92.2
10	82	5.55	94.4	28	4.47	90.4	26	2.97	88.2
25	55	4.48	93.6	19	3.68	91.7	17	2.53	86.7
50	45	3.99	92.7	16	3.58	91.9	14	2.33	85.7

Explicit Algorithm – Global Jump Stabilization ($\beta = 0.05$)									
	JPCG			SSOR-PCG			ESSOR-PCG		
No. of Vectors	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE
0	340	24.92	98.8	115	19.23	98.3	107	12.06	97.1
5	109	8.41	96.5	36	6.21	94.9	33	4.02	91.5
10	79	5.33	93.9	26	4.32	92.0	24	2.88	87.3
25	52	4.24	92.7	18	3.47	90.6	16	2.42	85.3
50	42	3.76	91.9	14	3.06	89.4	13	2.21	84.2

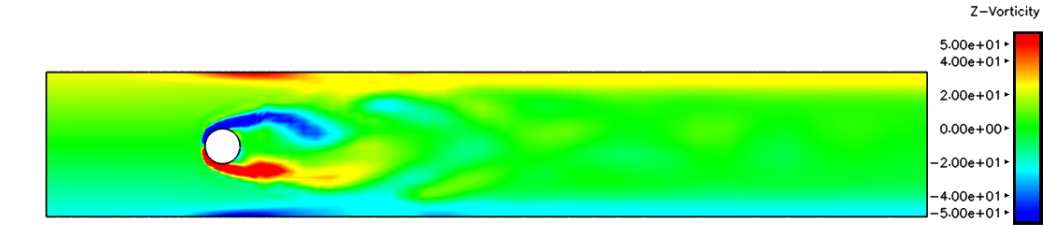
Explicit Algorithm – Local Jump Stabilization ($\beta = 0.05$)									
	JPCG			SSOR-PCG			ESSOR-PCG		
No. of Vectors	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE
0	291	21.44	98.7	99	16.55	98.1	91	10.35	96.8
5	120	9.27	97.0	40	6.84	95.5	36	4.31	92.5
10	89	5.96	94.7	30	4.80	93.0	27	3.19	88.7
25	62	4.96	94.2	21	3.98	92.1	19	2.73	87.4
50	50	4.31	93.1	17	3.46	90.8	15	2.47	86.1

Table 6.2: Effect of the A-Conjugate projection on the PPE solution time for the 2-D momentum driven jet. The grind times for the projection algorithm have been normalized with respect to the solution time for 1000 time steps using a direct solver [71, 72]. N_{IT} is the average number of iterations required per time step to solve the PPE problem. ($Nel = 11250$, $Nnp = 11466$, $N_b = 146$, direct solver grind time: 4.871×10^{-5} CPU seconds per element per time step)

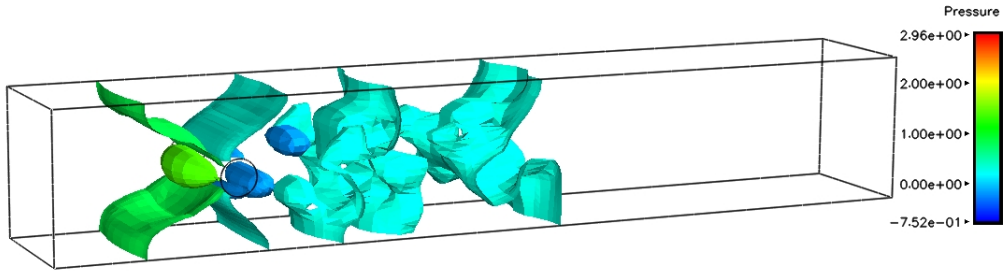
Projection-2 Algorithm – No Stabilization									
	JPCG			SSOR-PCG			ESSOR-PCG		
No. of Vectors	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE
0	328	6.52	87.4	110	5.05	83.6	100	3.36	74.7
5	148	3.20	75.0	49	2.63	69.4	47	2.02	59.7
10	121	2.53	66.5	41	2.13	59.9	38	1.75	51.1
25	100	2.66	69.4	34	2.18	62.3	31	1.71	51.7
50	87	2.39	66.6	29	2.03	60.4	26	1.60	49.2

Projection-2 Algorithm – Global Jump Stabilization ($\beta = 0.05$)									
	JPCG			SSOR-PCG			ESSOR-PCG		
No. of Vectors	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE
0	309	6.16	86.6	105	4.83	82.8	96	3.23	74.0
5	130	2.91	72.4	43	2.39	66.2	44	1.87	56.2
10	108	2.35	63.7	36	1.98	56.8	37	1.68	48.1
25	88	2.44	66.5	29	2.00	58.8	27	1.65	49.4
50	76	2.20	63.7	26	1.92	57.9	23	1.52	45.9

Projection-2 Algorithm – Local Jump Stabilization ($\beta = 0.05$)									
	JPCG			SSOR-PCG			ESSOR-PCG		
No. of Vectors	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE
0	309	6.52	86.6	105	4.95	82.8	96	3.27	74.0
5	130	3.11	72.4	43	2.53	66.2	44	1.93	56.2
10	108	2.47	63.7	36	2.07	56.8	37	1.73	48.1
25	88	2.61	66.5	29	2.09	58.8	29	1.69	49.4
50	76	2.36	63.7	26	2.04	57.9	25	1.58	46.0



a)



b)

Figure 6.6: Snapshot of a) cutplane showing the z-vorticity field and c) isosurfaces of the pressure for the 3-D cylinder in a channel for $Re_D = 100$.

the A-conjugate projection technique is to introduce both long and short wavelength information, thus effectively deflating the eigenvalue spectrum of the PPE. This effect not only reduces the iteration count and concomitant computational cost but is seen to improve the effectiveness of simple sub-domain parallel preconditioners that are effective for wavelengths proportional to the sub-domain size but cannot smooth error modes that span sub-domains.

Operational experience with both the global and local pressure stabilization formulations has shown that either approach is effective in terms of filtering spurious pressure modes and improving the overall robustness of the computations. Ultimately, the combination of the pressure stabilization with preconditioning and the A-conjugate projection technique has proven both robust and computationally efficient making it a reasonable alternative to more complicated approaches based on multi-grid or multi-level algorithms.

Table 6.3: Effect of the A-Conjugate projection on the PPE solution time for the 3-D square cylinder vortex shedding problem. The grind times for the explicit algorithm have been normalized with respect to the solution time for 1000 time steps using a direct solver [71, 72]. N_{IT} is the average number of iterations required per time step to solve the PPE problem. ($Nel = 22380$, $Nnp = 25568$, $N_b = 772$)

Explicit Algorithm – Global Jump Stabilization ($\beta = 0.05$)									
	JPCG			SSOR-PCG			ESSOR-PCG		
No. of Vectors	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE
0	191	9.75	97.2	67	8.00	96.5	59	5.17	94.3
5	109	5.82	95.3	37	4.57	93.9	35	3.22	91.2
10	92	4.97	94.5	30	3.89	92.8	27	2.66	89.2
25	65	3.66	92.3	21	2.86	90.0	19	1.96	85.5
50	49	2.91	90.4	16	2.33	87.7	14	1.66	82.7

Explicit Algorithm – Local Jump Stabilization ($\beta = 0.05$)									
	JPCG			SSOR-PCG			ESSOR-PCG		
No. of Vectors	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE
0	184	9.41	97.1	65	7.72	96.4	57	4.97	94.1
5	112	5.95	95.4	37	4.64	93.8	32	3.00	90.7
10	88	4.75	94.2	29	3.68	92.3	25	2.46	88.3
25	65	3.64	92.3	21	2.79	89.9	18	1.89	85.2
50	53	3.07	90.9	17	2.36	88.2	15	1.69	83.1

Table 6.4: Effect of the A-Conjugate projection on the PPE solution time for the 3-D square cylinder vortex shedding problem. The grind times for the projection algorithm have been normalized with respect to the solution time for 1000 time steps using a direct solver [71, 72]. N_{IT} is the average number of iterations required per time step to solve the PPE problem. ($Nel = 22380$, $Nnp = 25568$, $N_b = 772$)

Projection-2 Algorithm – Global Jump Stabilization ($\beta = 0.05$)									
	JPCG			SSOR-PCG			ESSOR-PCG		
No. of Vectors	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE
0	63	1.54	43.8	20	1.24	36.4	19	1.12	28.3
5	49	1.34	38.4	16	1.15	31.3	14	1.04	23.9
10	42	1.20	34.9	13	1.10	28.2	12	1.01	21.6
25	36	1.16	31.7	12	1.06	25.7	10	0.99	19.8
50	33	1.14	30.7	11	1.05	24.9	10	0.99	19.4

Projection-2 Algorithm – Local Jump Stabilization ($\beta = 0.05$)									
	JPCG			SSOR-PCG			ESSOR-PCG		
No. of Vectors	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE	N_{IT}	Grind Time	% PPE
0	67	1.43	45.0	21	1.26	37.6	20	1.12	29.5
5	51	1.30	39.0	16	1.16	32.1	15	1.05	24.7
10	44	1.10	35.8	14	1.11	28.9	13	1.02	22.3
25	39	1.06	32.8	12	1.08	26.6	11	1.00	20.5
50	36	1.05	31.9	11	1.07	25.8	10	0.99	20.1

Chapter 7

Turbulence Models

Turbulent flows occur in a wide variety of engineering applications and they are characterized by their inherent unpredictability, enhanced mixing properties and broad range of length and time scales. The length scales in a turbulent flow are bounded by the bulk or integral length scale, e.g., a pipe diameter or body dimension, and by the dissipation scale where physical viscosity acts. Similarly, the time scales are bounded by the time scale associated with the turn-over time of the largest eddies in the flow and the time scale for viscous dissipation. Resolution of all the length and time scales in high Reynolds numbers flows, i.e., direct numerical simulation, remains out of reach even for the largest supercomputers available today – particularly when complex geometry or complex physics is involved. For industrial flows, which are typically turbulent with relatively high Reynolds numbers, turbulence modeling is the pacing technology for computational fluid dynamics.

Turbulence models range in complexity from simple mixing-length or zero-equation models to n-equation models that provide second-moment closure. The n-equation models require the solution of additional partial differential equations that describe the transport of turbulence quantities, e.g., turbulent kinetic energy. The n-equation models range in complexity from one-equation models, e.g., the Spalart-Allmaras model [73] and the Baldwin-Barth model [74], to models that provide second-moment closure and require up to seven additional transport equations. Two-equation models are typified by the standard $k-\varepsilon$ model and its multitude of variations.

Large-eddy simulation is a promising alternative to traditional turbulence modeling approaches that rely upon Reynolds averaging and closure models based upon ad-hoc transport equations for the rate of dissipation. Unlike traditional Reynolds averaged models, LES provides a high degree of accuracy with minimal empiricism. However, the computational cost associated with performing *pure* LES, i.e., where a significant fraction of the energy spectrum is resolved, is quite high. For example, for simple channel flow, the grid resolution for *pure* LES scales approximately as Re^2 for simple channel flow (see for example [75, 76]). In contrast, the idea of under-resolved or VLES (very-large eddy simulation) has shown promising results in a variety of applications (see for example [77, 78, 79, 80, 81, 82]), suggesting that the non-purist view of LES may provide useful engineering results.

LS-DYNA's CFD capabilities are intended to find broad application, and since there is no single universal turbulence model that is suitable for all applications, the goal is to provide multiple turbulence models and modeling approaches. In the subsequent sections of this chapter, the averaging approaches, and current turbulence models available in LS-DYNA are outlined. Refer to Chapter 9 for details on the *CONTROL_CFD_TURBULENCE keyword which is used for the specification of turbulence model constants and model options.

7.1 Averaging, Filtering and Scale Separation

The broad range of length and time scales associated with the highly disordered, three-dimensional and time-dependent behavior of turbulent flow makes direct simulation for high-Reynolds number engineering flows computationally intractable. In order to attempt to model the behavior of turbulent flows, the fields are typically separated into mean and fluctuating components. The specific type of averaging used to obtain the mean flow fields depends on the flow regime and the type of turbulence model that is applied. The three types of averaging typically used in turbulence modeling are time, space and ensemble averaging.

Time averaging is used for flows that are considered stationary, i.e., the mean values are independent of the starting time for the averaging process. Stationary flows are also referred to as statistically steady, i.e., the mean values are not a function of time. For a generic flow variable, $\phi(x_i,t)$ where ϕ is taken to represent flow variables such as velocity, pressure, etc., the time average is

$$\overline{\phi}(x_i) = \lim_{T\to\infty} \frac{1}{T} \int_{t_0}^{t_0+T} \phi(x_i,t)\, dt \tag{7.1}$$

where $\overline{\phi}(x_i)$ is the averaged quantity and T is the duration for the averaging procedure. Note that in subsequent chapters, the average is also identified as $<\phi(x_i)>$ or $<\cdot>$. Using the average flow variable $\overline{\phi}$, the instantaneous value may be decomposed into average, $\overline{\phi}$, and fluctuating, ϕ', components as

$$\phi(x_i,t) = \overline{\phi}(x_i) + \phi'(x_i,t). \tag{7.2}$$

For flows that have a slowly varying mean that is not turbulent in nature, e.g., flows with an imposed mean periodic behavior, the time averaging procedure can be extended to account for the long and short time scales. In this situation, the time average is computed using Eq. (7.1) where $T_1 \ll T \ll T_2$, i.e., the time scale for the averaging must be large relative to the turbulent fluctuations and small relative to the time scale associated with the mean flow. As shown in Figure 7.1, T_1 is the time scale associated with the turbulent fluctuations, while T_2 is the time scale for the periodic mean flow. For this case, the decomposition becomes

$$\phi(x_i,t) = \overline{\phi}(x_i,t) + \phi'(x_i,t), \tag{7.3}$$

accounting for time-dependent mean flow variables $\overline{\phi}(x_i,t)$.

A homogeneous turbulent flow is one for which the flow field is uniform in all directions on the average. For homogeneous turbulent flows, a spatial average is appropriate where

$$\overline{\phi}(x_i) = \lim_{V\to\infty} \frac{1}{V} \int_{V} \phi(x_i,t)\, dV. \tag{7.4}$$

For flows that are not stationary, an ensemble average is appropriate. In this case, the average is computed over a significant number of flow trials until the mean quantities are independent of the number of trials. Formally, the ensemble average of is

$$\overline{\phi}(x_i,\tau) = \lim_{Ntrial\to\infty} \frac{1}{Ntrial} \sum_{n=1}^{Ntrial} \phi_n(x_i, t=\tau), \tag{7.5}$$

where $\tau = t - t_0$ is the elapsed time from the start of the flow experiment t_0, and $Ntrial$ is the number of experimental trials.

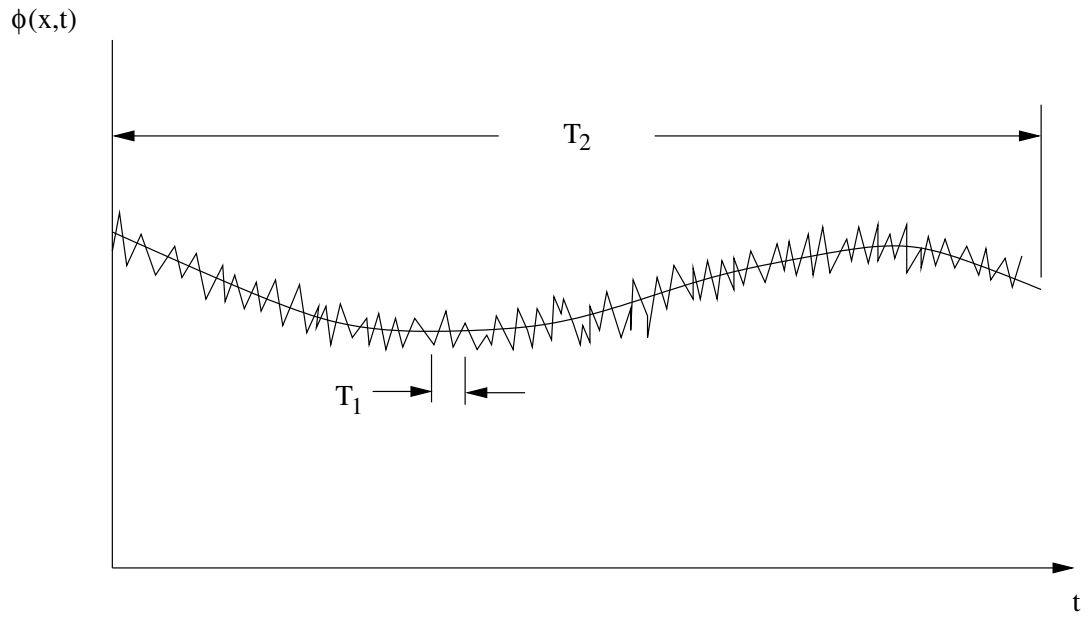


Figure 7.1: Time averaging for non-stationary flow.

For flows that are both stationary and homogeneous, time, space and ensemble averaging are all equivalent, i.e., the ergodic hypothesis applies. However, in a practical sense, homogeneous turbulent flow is strictly an idealization because boundary conditions such as at no-slip/no-penetration surfaces introduce flow inhomogeneities.

7.1.1 Spatial Filtering

An alternative approach to turbulence modeling is large-eddy simulation (LES) which relies on the direct computation of the largest flow structures with a model for the small-scale turbulence. The formulation of the LES problem relies on a separation of the flow field into resolved and sub-grid fields,

$$\phi(x_i,t) = \overline{\phi}(x_i,t) + \phi'(x_i,t), \tag{7.6}$$

where $\overline{\phi}$ is the resolved field and ϕ' is the sub-grid field. The resolved field is defined in terms of a spatial filter, which for homogeneous turbulence may be defined as

$$\overline{\phi}(x_i,t) = \int_\Omega \phi(\xi_i,t) G(x_i - \xi_i)\, d\Omega, \tag{7.7}$$

where G is the filter kernel and Ω is the flow domain. In order to reproduce constant fields, the filter must satisfy the condition that

$$\int_{\Omega^a} G(x_i)\, d\Omega = 1, \tag{7.8}$$

where Ω^a corresponds to the filter support. Examples of the Gaussian, sharp cut-off, and top-hat filters may be found in [75, 83].

The filtering procedures used for LES are different than the time averaging discussed above. Contrary to the traditional Reynolds time averaging,

$$\overline{\phi'} \neq 0, \tag{7.9}$$

and in general,

$$\overline{\overline{\phi}} \neq \overline{\phi}, \tag{7.10}$$

i.e., repeated filtering operations remove additional short wavelength information from the field.

The support for the filter indicated above by Ω^a introduces a length scale that corresponds to the smallest turbulence scale, Δ, that may be considered to be resolved. Thus, the spatial filtering procedure provides the mechanism to separate the flow variables into grid-resolved $\overline{\phi}$ and subgrid variations ϕ'.

7.2 The Time-Averaged Equations

In this section, the time-averaged form of the conservation equations are presented. As a starting point, the momentum equations and the divergence constraint are

$$\rho\frac{\partial u_i}{\partial t} + \rho u_j \frac{\partial u_i}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}(2\mu S_{ij}) + \rho f_i, \tag{7.11}$$

and

$$\frac{\partial u_i}{\partial x_i} = 0. \tag{7.12}$$

Here, the viscosity is treated as a scalar for simplicity, but internally LS-DYNA always treats the viscosity as a second-rank tensor for generality.

The time-averaged equations are derived by applying the additive decomposition $u_i = \overline{u}_i + u_i'$ and applying the time-averaging procedure from §7.1, where an over-bar indicates application of the time-averaging procedure in Eq. (7.3). The time-averaged momentum equations are

$$\rho\frac{\partial \overline{u}_i}{\partial t} + \rho\overline{u}_j \frac{\partial \overline{u}_i}{\partial x_j} = -\frac{\partial \overline{p}}{\partial x_i} + \frac{\partial}{\partial x_j}(2\mu\overline{S}_{ij} - \tau_{ij}) + \rho\overline{f}_i, \tag{7.13}$$

where

$$\overline{S}_{ij} = \frac{1}{2}\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right), \tag{7.14}$$

and the Reynolds stress tensor is

$$\tau_{ij} = -\overline{\rho u_i' u_j'}. \tag{7.15}$$

The divergence constraint is

$$\frac{\partial \overline{u}_i}{\partial x_i} = 0. \tag{7.16}$$

The Reynolds stress tensor is a symmetric second-rank tensor, $\tau_{ij} = \tau_{ji}$, with six independent components. Thus, six independent Reynolds stress components have been added to $\overline{u}_i$ and $\overline{p}$ as unknowns. However, there are only four equations (3-momentum equations and the divergence constraint) indicating that the system is not closed. In fact, the nonlinear terms in the momentum equation will yield new unknowns with each application of the averaging procedure making it impossible to generate a closed system.

In order to compute the time-averaged velocity and pressure, a prescription for the Reynolds stress is required. Ultimately, the purpose for turbulence models is to provide the prescription of the Reynolds stress and close the system of time-averaged equations, albeit with varying degrees of complexity and success.

7.3 The Spatially Filtered Equations

In this section, the spatially filtered form of the conservation equations are presented. Here, an over-bar indicates a spatially filtered quantity. Applying the spatial filtering procedure, Eq. (7.7), to Eq. (7.11) and (7.12) with the requirement that the filtering operation commutes with differentiation, the resulting space-filtered momentum equations are

$$\rho\frac{\partial\overline{u}_i}{\partial t}+\rho\overline{u}_j\,\frac{\partial\overline{u}_i}{\partial x_j}=-\frac{\partial\overline{p}}{\partial x_i}+\frac{\partial}{\partial x_j}(2\mu\overline{S}_{ij}-\tau_{ij}^{sgs})+\rho\overline{f}_i, \tag{7.17}$$

where the strain-rate based on the resolved velocity field is

$$\overline{S}_{ij}=\frac{1}{2}\left(\frac{\partial\overline{u}_i}{\partial x_j}+\frac{\partial\overline{u}_j}{\partial x_i}\right). \tag{7.18}$$

The divergence constraint is

$$\frac{\partial\overline{u}_i}{\partial x_i}=0. \tag{7.19}$$

Application of the spatial filter to the nonlinear advection term in the momentum equations produces a new term referred to as the subgrid-scale (SGS) stress tensor τ_{ij}^{sgs}. Using Germano's generalized moment definitions [84], the SGS stress tensor is

$$\tau_{ij}^{sgs}=\rho(\overline{u_iu_j}-\overline{u}_i\overline{u}_j), \tag{7.20}$$

and represents the effect of the subgrid-scale variations on the resolved scale. Like the Reynolds stress tensor, the SGS stress tensor is symmetric and introduces new unknowns which must be represented via a model. The definition of a model for the subgrid-scale stress tensor permits the solution of the filtered Navier-Stokes equations for the resolved-scale velocity $\overline{u}_i$ and pressure $\overline{p}$. However, like the time-averaged equations, closure of the spatially-filtered LES equations can vary in the degree of complexity and success in simulating turbulent flow.

The most simple and most common prescription for the SGS stresses relies on a parameterization in terms of the strain-rate and an eddy viscosity, i.e., the Boussinesq hypothesis. The Boussinesq hypothesis assumes that the principle axes of the SGS stress tensor, τ_{ij}^{sgs} are aligned with the principle axes of the filtered strain-rate tensor. Thus, the relationship between the deviatoric SGS stress and strain-rate is

$$\tau_{ij}^{sgs}-\frac{1}{3}\delta_{ij}\tau_{kk}^{sgs}=-2\mu_t\overline{S}_{ij}, \tag{7.21}$$

where μ_t is a turbulent eddy viscosity.

The parameterization of the deviatoric SGS stresses in terms of the strain-rate permits the filtered momentum equations to be written as

$$\rho\frac{\partial\overline{u}_i}{\partial t}+\rho\overline{u}_j\,\frac{\partial\overline{u}_i}{\partial x_j}=-\frac{\partial\hat{p}}{\partial x_i}+\frac{\partial}{\partial x_j}\left\{2(\mu+\mu_t)\overline{S}_{ij}\right\}+\rho\overline{f}_i, \tag{7.22}$$

where

$$\hat{p}=\overline{p}+\frac{1}{3}\tau_{kk}^{sgs}. \tag{7.23}$$

7.3.1 Baseline Smagorinsky Subgrid-Scale Model

One of the most common models for the LES subgrid-scale stresses was first proposed by Smagorinsky [85] and uses the Boussinesq hypothesis to relate the deviatoric part of the SGS stress tensor to the strain-rate via an eddy viscosity. The basis for the Smagorinsky model lies in the observation that the turbulent eddy-viscosity can be parameterized in terms of a length and a velocity scale. This observation permits the eddy-viscosity to be written in terms of a length scale and the rate of dissipation, ε, as

$$\mu_t = C\Delta^{4/3}\varepsilon^{1/3}, \tag{7.24}$$

where the length scale has been chosen as the filter scale, Δ.

Assuming that the dissipation rate is in equilibrium with the rate of production of subgrid kinetic energy, $P^{sgs} = -\tau_{ij}^{sgs}\overline{S}_{ij}$, permits estimation of the eddy-viscosity in terms of the filtered strain-rate. In the Smagorinsky model, the turbulent eddy viscosity is

$$\mu_t = \rho(C_S\Delta)^2\sqrt{\overline{S}_{ij}\overline{S}_{ij}}. \tag{7.25}$$

The Smagorinsky *constant*, C_S, is not universal and requires adjustment for each unique set of flow conditions. Various applications have shown that C_S typically is in the range of $0.1 \leq C_S \leq 0.24$ (see Rogallo and Moin [86]. In LS-DYNA, the default for the baseline Smagorinsky model is $C_S = 0.1$ (see the *CONTROL_CFD_TURBULENCE keyword in Chapter 9).

7.4 Subgrid Kinetic Energy Models

The Smagorinsky model has found broad application and is attractive from a computational perspective due to its simplicity. While the Smagorinsky model provides approximately the correct amount of dissipation to the resolved velocity field, the model has several deficiencies. For example, the Smagorinsky constant is not universal, the eddy viscosity does not vanish in laminar regions, the limiting wall behavior is incorrect, and the model is excessively dissipative for transitional flows.

This section outlines the LES models in LS-DYNA that are based on the subgrid-scale kinetic energy. The SGS kinetic energy is

$$k^{sgs} = \frac{1}{2}(\overline{u_k u_k} - \overline{u}_k\overline{u}_k). \tag{7.26}$$

Following Menon and Kim [87, 88], the transport equation for the SGS kinetic energy is

$$\frac{\partial k^{sgs}}{\partial t} + \overline{u}_i\frac{\partial k^{sgs}}{\partial x_i} = -\tau_{ij}^{sgs}\frac{\partial \overline{u}_i}{\partial x_j} - \varepsilon^{sgs} + \frac{\partial}{\partial x_i}\left(\frac{\mu_t}{\rho}\frac{\partial k^{sgs}}{\partial x_i}\right), \tag{7.27}$$

where ε^{sgs} is the subgrid-scale dissipation. This transport equation is based on the difference between the equation of mechanical energy and its spatially-filtered form, and it is similar to the k-transport equation of Schumann [89] and follows the form used by Yoshizawa [90].

In Eq. (7.27), the SGS stresses are modeled using the Boussinesq hypothesis with the filter scale, Δ, providing the length scale and the SGS kinetic energy providing the velocity scale. Thus, the SGS stresses are

$$\tau_{ij}^{sgs} = -2C_\tau \Delta \sqrt{k^{sgs}}\, \overline{S}_{ij} + \frac{1}{3}\delta_{ij}\tau_{kk}^{sgs}, \tag{7.28}$$

where $\tau_{kk}^{sgs} = 2k^{sgs}$, and the eddy viscosity is

$$\mu_t = \rho C_\tau \Delta \sqrt{k^{sgs}}. \tag{7.29}$$

In order to close Eq. (7.27), the dissipation rate is modeled as

$$\varepsilon^{sgs} = C_\varepsilon \frac{(k^{sgs})^{3/2}}{\Delta}. \tag{7.30}$$

In order to solve Eq. (7.27), the constants C_τ and C_ε must be prescribed . Schmidt and Schumann [91] suggest $C_\tau = 0.0856$ and $C_\varepsilon = 0.845$. Alternatively, Chakravarthy and Menon [92] have determined that $C_\tau = 0.067$ and $C_\varepsilon = 0.931$ based on the an analysis of the inertial-viscous spectrum and SGS closure.

Unlike the Smagorinsky model which relates the eddy viscosity to the instantaneous filtered strain-rate, the k^{sgs} model includes some effects of the flow history in the estimation of the eddy viscosity. In addition, there are no explicit assumptions equating the production and dissipation of turbulent kinetic energy making it possible to account for non-equilibrium effects. However, this model still requires the use of flow-specific constants. In the following section, the dynamic procedure for evaluating the constants is outlined.

7.4.1 Local Dynamic k^{sgs} model

The local dynamic k^{sgs} model was was first proposed by Kim and Menon [93] and later extended to compressible flows by Nelson [94]. The *dynamic* evaluation of C_τ and C_ε makes these quantities functions of both space and time. The dynamic evaluation procedure (see for example [95, 96]) relies on a test-scale or high-pass filter with a filter scale $\widehat{\Delta} > \Delta$ where typically $\widehat{\Delta}/\Delta \approx 2$. In the ensuing discussion, a circumflex, " $\widehat{\ }$ ", denotes application of the test-scale filter.

Application of the test-scale filter further decomposes the flow variables into test-scale and subtest-scale components, e.g., the subtest stresses are

$$T_{ij} = \rho(\widehat{\overline{u_i}\overline{u_j}} - \widehat{\overline{u}_i}\widehat{\overline{u}_j}). \tag{7.31}$$

Application of the test-scale filter to the subgrid-scale stresses yields a relationship between the test-scale and SGS stresses,

$$L_{ij} = \rho(\widehat{\overline{u_i}\overline{u_j}} - \widehat{\overline{u}_i}\widehat{\overline{u}_j}). \tag{7.32}$$

where $L_{ij} = T_{ij} - \widehat{\tau}_{ij}^{sgs}$.

The test-scale kinetic energy is defined as the trace of Eq. (7.32), i.e.,

$$k^{test} = \frac{1}{2}L_{kk}. \tag{7.33}$$

The test-scale kinetic energy is analogous to k^{sgs}, but it is produced at the large scales and dissipated at the small scales by

$$e = \frac{(\mu+\mu_t)}{\rho}\left(\widehat{\overline{\frac{\partial u_i}{\partial x_j}\frac{\partial u_i}{\partial x_j}}} - \frac{\partial \hat{\bar{u}}_i}{\partial x_j}\frac{\partial \hat{\bar{u}}_i}{\partial x_j}\right). \tag{7.34}$$

Similarly, the dissipation is based on the molecular and turbulent viscosity since k^{test} represents the resolved kinetic energy at the test-scale.

The SGS stresses, τ_{ij}^{sgs}, and the Leonard stresses, L_{ij}, have been observed to exhibit similarities in experimental data as reported by by Liu et al. [97] for the fully developed region of a circular jet. This suggests that L_{ij} be modeled in a manner similar to the SGS stresses, that is,

$$L_{ij} = -2C_\tau\widehat{\Delta}\sqrt{k^{test}}\,\widehat{\bar{S}}_{ij} + \frac{1}{3}\delta_{ij}L_{kk}. \tag{7.35}$$

Here, C_τ is the only unknown since L_{ij} may be obtained by application of the test-scale filter to the computed velocities. Thus, Eq. (7.35) provides an explicit representation for C_τ in terms of known quantities. However, this equation is overdetermined because there are five equations and only a single unknown, C_τ. Using the least-squares method introduced by Lilly [96],

$$C_\tau = \frac{L_{ij}\sigma_{ij}}{\sigma_{ij}\sigma_{ij}}, \tag{7.36}$$

where

$$\sigma_{ij} = -\hat{\Delta}\sqrt{k^{test}}\,\hat{\bar{S}}_{ij}, \tag{7.37}$$

which can be determined entirely from data available at the test scale.

In comparison to Germano's dynamic procedure which can exhibit pathalogical behavior requiring additional spatial averaging in homogeneous directions, the evaluation of C_τ in Eq. (7.36) is well-defined and non-zero ($\sigma_{ij}\sigma_{ij} \neq 0$). In addition, in this dynamic procedure, there is no requirement for the model constant C_τ to be filter independent, i.e., the evaluation is strictly local. Thus, the model is referred to as the local dynamic k^{sgs} model (LDKM).

In order to obatin the dissipation model constant, C_ε, similarity between e and ε is assumed. Thus,

$$e = \frac{C_\varepsilon}{\hat{\Delta}}(k^{test})^{3/2}, \tag{7.38}$$

which permits evaulation of the dissipation constant as

$$C_\varepsilon = \frac{(\mu+\mu_t)}{\rho}\frac{\hat{\Delta}}{(k^{test})^{3/2}}\left(\widehat{\overline{\frac{\partial u_i}{\partial x_j}\frac{\partial u_i}{\partial x_j}}} - \frac{\partial \hat{\bar{u}}_i}{\partial x_j}\frac{\partial \hat{\bar{u}}_i}{\partial x_j}\right). \tag{7.39}$$

Chapter 8

Flow Statistics

The purpose for this chapter is to provide sufficient information for an experienced analyst to use LS-DYNA to generate flow statistics from a transient flow simulation. The goal of this chapter is to present the "built-in" tools that are available in LS-DYNA and LS-POST to analyze time-dependent flow data, specifically turbulent flow data. It is through the use of these tools that the user will develop an understanding of the simulated flow field which may lead to design changes or implementation of turbulence controls.

LS-DYNA provides the capability to accumulate time-averaged data that may be later post-processed with LS-POST to derive flow statistics. For large-eddy simulations, turbulence statistics such as the mean and fluctuating velocity, Reynolds stresses, turbulent kinetic energy, velocity–temperature and velocity–pressure correlations and higher-order statistics such as velocity skewness and flatness may be generated. LS-DYNA generates a "time-averaged" database (the *d3mean* file) that may be used with LS-POST which provides the capability to select a time-averaging window for the generation of all the derived turbulence statistics. In this model, LS-DYNA generates the raw statistical data that is analyzed by LS-POST to generate flow statistics.

In §8.1, a theoretical overview is presented as background while §8.2 summarizes the statistics level generated by LS-DYNA and §8.3 presents the derived statistics available in LS-POST.

Note that LS-POST support for *d3mean* and *d3thins* files will only be available in LS-POST 2.0 and later versions. *d3thins* is the new time history database for the incompressible flow solver.

8.1 Derived Flow Statistics

Some of the most important statistical quantities of interest (and the most physically relevant) are the mean values of the flow variables, sometimes referred to as moments. The first two moments are the mean and variance. These moments are routinely measured in experiments and form the basis of the Reynolds Averaged Navier-Stokes (RANS) solution approach for turbulent flows. There are several techniques used to obtain these moments from raw data. Before proceeding with an overview of the methods used to derive statistics from a time-accurate flow simulation, several definitions are required.

8.1.1 Reynolds Averaged Statistics

Reynolds time-averages, homogeneous spatial averages, and ensemble averages are three types of averaging commonly used to determine mean quantities (or moments) of a turbulent flow. Reynolds averaging describes the time invariant mean quantities of a stationary flow as a function of position. Ensemble averages are well suited for transient flows and describe the mean values as a function of both space and elapsed time. Spatial averages describe the spatial invariance of mean values in a homogeneous flow that may or may not be stationary.

A stationary flow is one for which the mean values are independent of the starting time for the averaging process. Stationary flows are also referred to as statistically steady, i.e., the mean values are not a function of time. For a generic flow variable, $\phi(x_i,t)$ where $\phi = u,v,w,p,\ldots$, etc., the Reynolds time-average is

$$< \phi(x_i) >= \lim_{T\to\infty} \frac{1}{T} \int_{t_0}^{t_0+T} \phi(x_i,t)\, dt \tag{8.1}$$

where $< \phi(x_i) >$ is the averaged quantity and T is the duration for the averaging process as shown in Figure 8.1.

Remark

> Note that the time-average was defined in an equivalent way in Chapter 7, and was referred to as $\overline{\phi}(x_i)$. Here, $< \cdot >$ is used because this notation is directly translated into the graphics database and LS-POST.

In a discrete sense, the time-average is approximated by

$$< \phi(x_i) >= \lim_{N\to\infty} \frac{\sum_{n=1}^{N} \phi^n(x_i,t^n)\Delta t^n}{\sum_{n=1}^{N} \Delta t^n} \tag{8.2}$$

where Δt^n is the time step increment for the nth time integration step and $t^n = \sum_{m=1}^{n} \Delta t^m$ is the accumulated time.

A flow is considered to be non-stationary if there is no time invariant mean that can be defined. In order to define mean quantities for a non-stationary flow an ensemble average is required. Here, the mean values are computed by averaging over a significant number of flow trials until the mean quantities are independent of the number of trials. Formally, the ensemble average of $\phi(x_i,t)$ is

$$< \phi(x_i,\tau) >= \lim_{Ntrial\to\infty} \frac{1}{Ntrial} \sum_{n=1}^{Ntrial} \phi_n(x_i, t=\tau), \tag{8.3}$$

where $\tau = t - t_0$ is the elapsed time from the start of the flow experiment t_0, and $Ntrial$ is the number of computational experiments.

In a spatially homogeneous flow, the statistical properties of the flow do not vary in space. All the mean values are independent of location and can by described by a single value at an instant in time, instead of a value at each spatial location. The mean values for homogeneous flows are determined from a volume average, i.e.,

$$\overline{\phi(t)} = \lim_{\Omega\to\infty} \frac{1}{\Omega} \int_{\Omega} \phi(x_i,t) d\Omega, \tag{8.4}$$

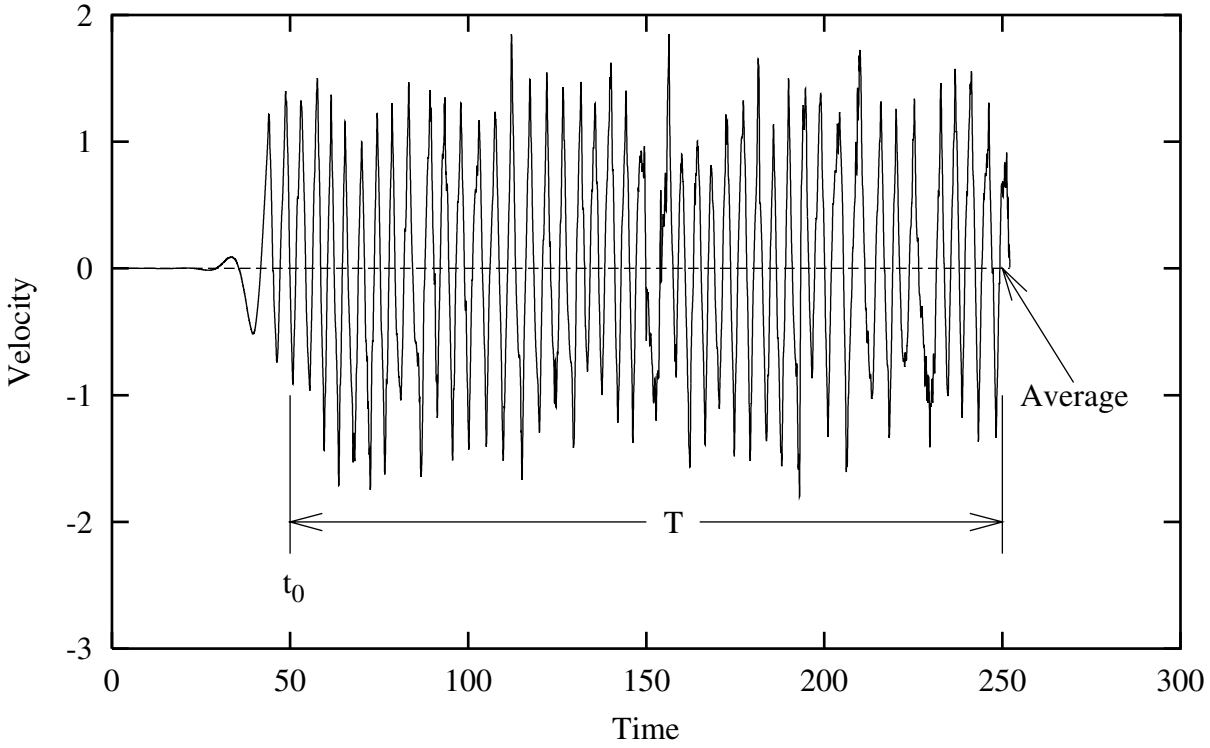


Figure 8.1: Time-averaged and instantaneous velocity.

where Ω is the volume of the domain. Of course, homogeneous flow is an idealization because boundary conditions such as no-slip/no-penetration surfaces introduce flow inhomogeneities.

For flows that are both stationary and homogeneous, all three forms of averaging are equivalent, i.e., the ergodic hypothesis applies. The current version of LS-POST calculates Reynolds averaged statistics of stationary flows. Spatial averaging of homogeneous data, ensemble averaging of transient flow data and calculation of probability densities have been neglected in the current version of the software.

8.1.2 Mean and Fluctuating Quantities

Turning now to the calculation of mean and fluctuating components, the definition of the Reynolds average in Eq. (8.1) permits any variable can be decomposed into its mean and fluctuating component,

$$\phi(x_i,t) = <\phi(x_i)> + \phi'(x_i,t), \tag{8.5}$$

where $<\phi>$ is the mean and ϕ' is the fluctuating part. Given the definition of the Reynolds average, application of the average to the fluctuating component yields

$$<\phi'> = 0. \tag{8.6}$$

Repeated application of the averaging procedure does not change the average,

$$<<\phi>> = <\phi>. \tag{8.7}$$

Averaging a product of variables, i.e., ϕ and ψ yields

$$<\phi\psi> = <\phi><\psi> + <\phi'\psi'>. \tag{8.8}$$

Similarly, the averaging procedure is commutative and averaging with a fluctuating component yields

$$<\phi' <\psi>>=<\psi' <\phi>>= 0. \tag{8.9}$$

Let ϕ represent a scalar quantity and let u_i, $(i = 1,2,3)$ represent the ith Cartesian component of the instantaneous velocity.

For a scalar quantity, ϕ, the variance is

$$\sigma^2 =<\phi^2> - <\phi>^2, \tag{8.10}$$

and the root mean square (RMS) value is σ. The triple correlation between three independent scalars, ϕ, ψ, and λ is

$$\begin{aligned} <\phi'\psi'\lambda'> &= <\phi\psi\lambda> - <\phi><\psi><\lambda> \\ &\quad -(<\phi><\psi'\lambda'> + <\psi><\phi'\lambda'> + <\lambda><\phi'\psi'>). \end{aligned} \tag{8.11}$$

Correlations of this type appear in second order moment closure models such as the Reynolds stress turbulence model. Higher order correlations can be defined in a similar manner.

The Reynolds stress tensor is defined in a similar way

$$R_{ij} =<u_i'u_j'>=<u_iu_j> - <u_i><u_j>, \tag{8.12}$$

where u_i are the velocity components. The turbulent kinetic energy is

$$q' = \frac{1}{2} <u_k'u_k'>= \frac{1}{2}(<u_ku_k> - <u_k><u_k>). \tag{8.13}$$

The turbulent scalar flux vector is

$$<\phi'u_i'>=<\phi u_i> - <\phi><u_i> . \tag{8.14}$$

8.1.3 Higher-Order Statistics

Higher-order moments of velocity differences and velocity derivatives are frequently used to analyze isotropic turbulence. The skewness is defined as

$$S_{0i} = \frac{\left\langle u_i^3 \right\rangle}{\left\langle u_i^2 \right\rangle^{3/2}}, \tag{8.15}$$

and the flatness factor is

$$F_{0i} = \frac{\left\langle u_i^4 \right\rangle}{\left\langle u_i^2 \right\rangle^{2}}. \tag{8.16}$$

The skewness is a measure of the asymmetry of the fluctuations; $S_{0i} > 0$ are predominately positive and $S_{0i} < 0$ are predominately negative. The flatness is a relative measure of remotely occurring, symmetric spiking fluctuations.

The skewness of velocity derivatives is defined as

$$S_{1i} = \frac{\left\langle \left(\frac{\partial u_i}{\partial x_i}\right)^3 \right\rangle}{\left\langle \left(\frac{\partial u_i}{\partial x_i}\right)^2 \right\rangle^{3/2}}, \tag{8.17}$$

and the flatness factor of velocity derivatives

$$F_{1i} = \frac{\left\langle \left(\frac{\partial u_i}{\partial x_i}\right)^4 \right\rangle}{\left\langle \left(\frac{\partial u_i}{\partial x_i}\right)^2 \right\rangle^2} \tag{8.18}$$

where there is no summation of repeated indices.

Skewness and flatness of velocity differences have been also measured [98],

$$S_{\Delta i}(r) = \frac{\left\langle (\Delta u_i)^3 \right\rangle}{\left\langle (\Delta u_i)^2 \right\rangle^{3/2}} \tag{8.19}$$

and the flatness factor is

$$F_{\Delta i}(r) = \frac{\left\langle (\Delta u_i)^4 \right\rangle}{\left\langle (\Delta u_i)^2 \right\rangle^2} \tag{8.20}$$

where $\Delta u_i = u_i(x_i + r_i, t) - u_i(x_i, t)$ and $r_i = x_i + r$.

For homogeneous and isotropic turbulence, the velocity fluctuations have a Gaussian probability distribution. Since this distribution is symmetric, all odd moments are zero. The nonzero moment higher than the variance is the flatness that obtains a value of 3.0. The derivative skewness is $\sim -0.3 - -0.5$ and the derivative flatness is $\sim 3 - 4$ [98]. Velocity derivative skewness and flatness are dominated by the smaller scales in the flow so these statistics provide a good measure of how well the numerical scheme is resolving the smaller scales of the flow.

8.1.4 Anisotropic Stress Tensor

An intrinsic distinction exists between isotropic and anisotropic Reynolds stresses. The isotropic stress is $\frac{2}{3}q'\delta_{ij}$ and the deviatoric anisotropic part is

$$a_{ij} = < u_i' u_j' > - \frac{2}{3} q' \delta_{ij}. \tag{8.21}$$

The normalized anisotropy tensor is defined as $b_{ij} = \frac{a_{ij}}{2q'}$. It is readily seen that $a_{kk} = 0$ and any anisotropy is seen by non-zero values of the off-diagonal terms.

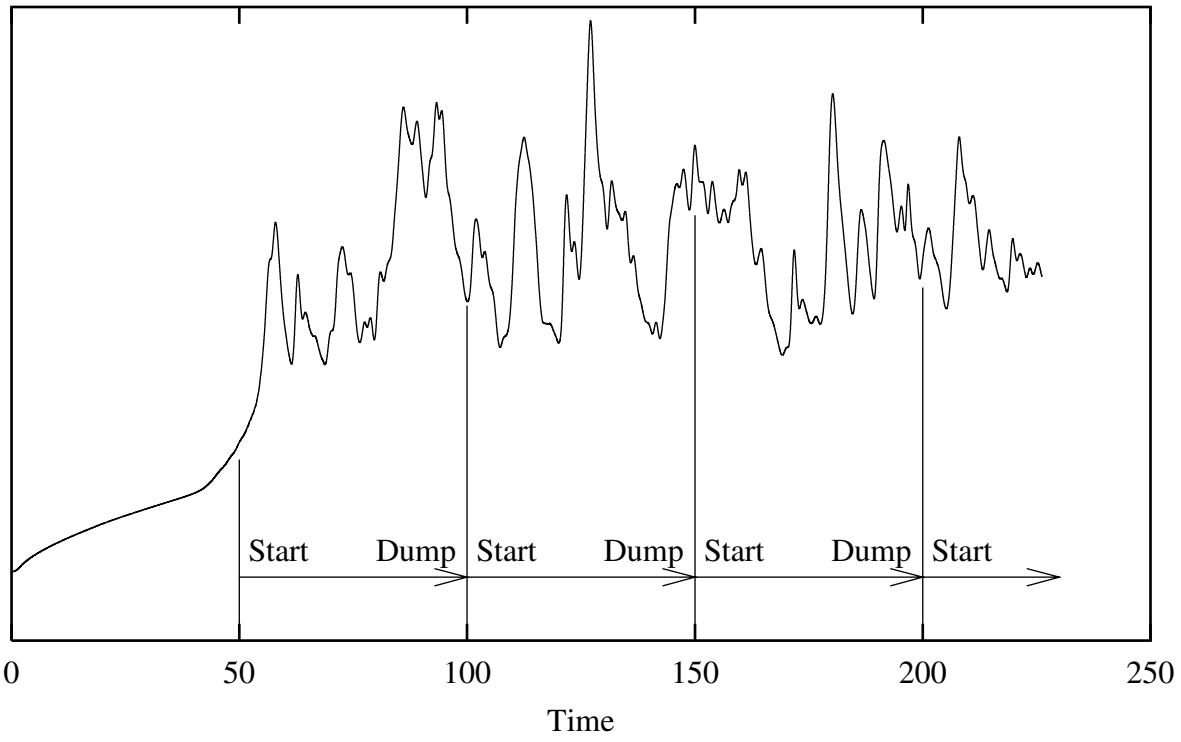


Figure 8.2: Time windows for the collection of the raw time-averaged data for LS-POST.

8.2 LS-DYNA Statistics Levels

Currently, LS-DYNA generates three levels of time-averaged output data that range from mean quantities (level-1) such as mean velocity, pressure, etc. to an intermediate level (level-2) that includes correlations such as velocity and pressure, and the highest level (level-3) that includes the higher-order correlations, $< u_i^3 >$ and $< u_i^4 >$. The level of time-averaging is selected using the *ISTATS* parameter on the *DATABASE_BINARY_D3MEAN keyword.

At run-time, LS-DYNA generates the *d3mean* data file that contains raw time-averaged data that is collected during the course of a simulation. In order to facilitate the computation of a windowed time-average for the derived statistics, LS-DYNA collects the raw time-averaged data over a series of time windows as shown in Figure 8.2. Here, the time averaging procedure starts at $t_{start} = 50$ and is restarted after the data for a time window is written to disk. This approach permits the exclusion of start-up transients in the computation of the derived turbulence statistics. The starting time for the collection of data in LS-DYNA is controlled by the *TSTART* parameter on the *DATABASE_BINARY_D3MEAN keyword.

For each time-window, the LS-DYNA collects the raw statistics for the active time window. Considering a generic variable, $\phi(x_i,t)$, the time-average for a given window is simply

$$< \phi(x_i) >= \frac{1}{t^{n+Iavg} - t^n} \sum_{j=n}^{n+Iavg} \phi(x_i,t^j)\Delta t^j, \tag{8.22}$$

where *Iavg* is the number of time samples accumulated in the window.

LS-DYNA records the time associated with the end of each time window, the elapsed time associated the window, and the number of samples collected in the window. The elapsed time, $t^{n+Iavg} - t^n$, and the number of time samples during this time interval, *Iavg*, is written into the *d3mean* database. The number of time samples in a window is controlled by the *IAVG* parameter on the *DATABASE_BINARY_D3MEAN keyword.

8.2.1 Level-1: Mean Statistics

For level-1 time-averaged statistics, LS-DYNA accumulates the variables shown in Table 8.1 with their associated database labels for the *d3mean* file. In this statistics level, derived mean quantities that are based on the user-specified time window include: the velocity, $< u_i >$, pressure, $< p >$, vorticity, $< \omega_i >$, and for 2D problems, the vorticity, $< \omega >$, and the stream function, $< \psi >$. For thermal problems, the mean quantities also include the average temperature, $< T >$, and for species transport, the mean species concentration, $< Z_j >$.

Variable Name	Quantity	LS-POST Label
X-velocity	$< u_1 >$	<X-Velocity>
Y-velocity	$< u_2 >$	<Y-Velocity>
Z-velocity	$< u_3 >$	<Z-Velocity>
Pressure	$< p >$	<p>
X-vorticity	$< \omega_1 >$	<X-Vorticity>
Y-vorticity	$< \omega_2 >$	<Y-Vorticity>
Z-vorticity	$< \omega_3 >$	<Z-Vorticity>
Vorticity (2D)	$< \omega >$	<vorticity>
Stream Function	$< \psi >$	<stream>
Temperature	$< T >$	<T>
Species-1	$< Z_1 >$	<Z1>
Species-2	$< Z_2 >$	<Z2>
Species-3	$< Z_3 >$	<Z3>
Species-4	$< Z_4 >$	<Z4>
Species-5	$< Z_5 >$	<Z5>
Species-6	$< Z_6 >$	<Z6>
Species-7	$< Z_7 >$	<Z7>
Species-8	$< Z_8 >$	<Z8>
Species-9	$< Z_9 >$	<Z9>
Species-10	$< Z_{10} >$	<Z10>

Table 8.1: LS-DYNA **ISTATS=**1 output variables and LS-POST labels.

8.2.2 Second Moment Statistics

The computation of the second-moment statistics includes the variables listed in Table 8.2. This data, as well as the variables listed in Table 8.1, must be present in the input time-average database (*d3mean*) in order for the statistics associated with **ISTATS =** 2 to be computed. Therefore, selection of the level-2 statistics in LS-DYNA includes the collection of all the level-1 time-averaged data.

Variable Name	Quantity	LS-POST Label
Velocity Correlations	$< u_1u_1 >$	<u1u1>
	$< u_1u_2 >$	<u1u2>
	$< u_1u_3 >$	<u1u3>
	$< u_2u_2 >$	<u2u2>
	$< u_2u_3 >$	<u2u3>
	$< u_3u_3 >$	<u3u3>
Velocity – Pressure Correlations	$< u_1p >$	<u1p>
	$< u_2p >$	<u2p>
	$< u_3p >$	<u3p>
Velocity – Temperature Correlations	$< u_1T >$	<u1T>
	$< u_2T >$	<u2T>
	$< u_3T >$	<u3T>
Velocity – Species Correlations	$< u_1Z_1 >$	<u1Z1>
	$< u_2Z_1 >$	<u2Z1>
	$< u_3Z_1 >$	<u3Z1>
	$< u_1Z_2 >$	<u1Z2>
	$< u_2Z_2 >$	<u2Z2>
	$< u_3Z_2 >$	<u3Z2>
	.	
	.	
	.	
	$< u_1Z_{10} >$	<u1Z10>
	$< u_2Z_{10} >$	<u2Z10>
	$< u_3Z_{10} >$	<u3Z10>

Table 8.2: LS-DYNA **ISTATS=**2 output variables and LS-POST labels.

8.2.3 Higher-Order Statistics

The computation of the higher-order statistics includes the variables listed in Table 8.3. This data, along with the variables listed in Table 8.2, must be present in the input time-average database in order for the statistics associated with **ISTATS =** 3 to be computed.

Variable Name	Quantity	LS-POST Label
Third Moments	$< u_1^3 >$	<u1^3>
	$< u_2^3 >$	<u2^3>
	$< u_3^3 >$	<u3^3>
Fourth Moments	$< u_1^4 >$	<u1^4>
	$< u_2^4 >$	<u2^4>
	$< u_3^4 >$	<u3^4>

Table 8.3: LS-DYNA **ISTATS=**3 output variables and LS-POST labels.

Variable Name	Quantity	LS-POST Label
Mag. vorticity	$\| < \omega > \|$	|<vorticity>|
Enstrophy	$1/2 < \omega > \cdot < \omega >$	<enstrophy>
Helicity	$< u > \cdot < \omega >$	<helicity>

Table 8.4: LS-POST level-1 derived variables and LS-POST labels.

8.3 LS-POST Statistics Levels

The derived statistics generated by LS-POST from the *d3mean* database are also provided in three levels. The derived statistics are based on the same user-specified time-average window used to create each *d3mean* time-average data window. Hence, this derived data is also a subset of the input data from the LS-DYNA flow simulation. The level-1 derived mean quantities include: the magnitude of the vorticity, $\| < \omega > \|$, enstrophy, $1/2 < \omega_i > \cdot < \omega_i >$, and helicity, $< \omega_i > \cdot < u_i >$.

At the intermediate level of derived statistics (level-2), second moments are computed in addition to the mean quantities just mentioned. In this case, the turbulent stress tensor and scalar flux vectors are computed. That is, $< u_i' u_j' >$, $< u_i' p' >$, $< u_i' T' >$, $< u_i' Z_j' >$, and the turbulent kinetic energy are added to the output time-average database.

At the highest level of statistics, skewness and flatness of the velocity are computed in addition to the mean and derived flux quantities. Currently, the scalar dissipation is not treated in LS-POST.

Thus, the time-averaged statistics derived by LS-POST from the *d3mean* database are:

level-1: the magnitude of the vorticity, $\| < \omega > \|$, enstrophy, $1/2 < \omega_i > \cdot < \omega_i >$, and helicity, $< \omega_i > \cdot < u_i >$. Table 8.4 shows these mean variables and the associated LS-POST labels.

level-2: In addition to the mean quantities for **level-1**, the turbulent stress tensor, turbulent kinetic energy, and scalar flux vectors are computed. That is, $< u_i' u_j' >$, q', $< u_i' p' >$, $< u_i' T' >$, $< u_i' Z_j' >$, are added to the time-average database. Table 8.5 shows the additional output variables associated with **level-2**.

level-3: In addition to the mean and derived flux quantities, higher-order averages are generated that include velocity skewness and velocity flatness. Table 8.6 shows the additional output variables associated with **level-3**. At this time, the scalar dissipation is not included in the time-average database.

Variable Name	Quantity	LS-POST Label
Reynolds Stresses	$< u_1' u_1' >$	<u1'u1'>
	$< u_1' u_2' >$	<u1'u2'>
	$< u_1' u_3' >$	<u1'u3'>
	$< u_2' u_2' >$	<u2'u2'>
	$< u_2' u_3' >$	<u2'u3'>
	$< u_3' u_3' >$	<u3'u3'>
Turbulent Kinetic Energy	q'	TKE
Velocity – Pressure Correlations	$< u_1' p' >$	<u1'p'>
	$< u_2' p' >$	<u2'p'>
	$< u_3' p' >$	<u3'p'>
Velocity – Temperature Correlations	$< u_1' T' >$	<u1'T'>
	$< u_2' T' >$	<u2'T'>
	$< u_3' T' >$	<u3'T'>
Velocity – Species Correlations	$< u_1' Z_1' >$	<u1'Z1'>
	$< u_2' Z_1' >$	<u2'Z1'>
	$< u_3' Z_1' >$	<u3'Z1'>
	$< u_1' Z_2' >$	<u1'Z2'>
	$< u_2' Z_2' >$	<u2'Z2'>
	$< u_3' Z_2' >$	<u3'Z2'>
	.	
	.	
	.	
	$< u_1' Z_{10}' >$	<u1'Z10'>
	$< u_2' Z_{10}' >$	<u2'Z10'>
	$< u_3' Z_{10}' >$	<u3'Z10'>

Table 8.5: LS-POST level-2 derived variables and LS-POST Labels.

Variable Name	Quantity	LS-POST Label
Velocity Skewness	$< u_1^3 > / < u_1^2 >^{3/2}$	<X-skewness>
	$< u_2^3 > / < u_2^2 >^{3/2}$	<Y-skewness>
	$< u_3^3 > / < u_3^2 >^{3/2}$	<Z-skewness>
Velocity Flatness	$< u_1^4 > / < u_1^2 >^2$	<X-flatness>
	$< u_2^4 > / < u_2^2 >^2$	<Y-flatness>
	$< u_3^4 > / < u_3^2 >^2$	<Z-flatness>

Table 8.6: LS-POST level-3 derived variables and LS-POST labels.

Chapter 9

Keyword Input

This chapter outlines the primary keywords used to control the incompressible flow solver and discusses how these keywords interact with LS-DYNA's keywords for solid/structural dynamics, thermal analysis, boundary element and smooth particle hydrodynamics keywords. This chapter is intended to be used in conjunction with the LS-DYNA Keyword User's Manual [27] which fully documents the keywords and their associated parameters.

Note that LS-POST support for *d3mean* and *d3thins* files will only be available in LS-POST 2.0 and later versions. *d3thins* is the new time history database for the incompressible flow solver. Selecting what data goes into the *d3mean* database is accomplished via the *DATABASE_BINARY_D3MEAN keyword, documented in the LS-DYNA Keyword User's Manual.

9.1 Incompressible Flow Keywords

In LS-DYNA, node sets are used to prescribe nodal boundary conditions, while segment sets are used to apply known flux and traction boundary conditions. The use of node and segment sets follows the conventions defined in the LS-DYNA Keyword User's Manual [27].

9.1.1 *BOUNDARY_OUTFLOW_CFD_*OPTION***

Available options include: **SET** or **SEGMENT**

Purpose: Define *passive* outflow boundary conditions for the incompressible flow solvers. These boundary conditions are active only when SOLN=4 or SOLN=5 on the *CONTROL_SOLUTION keyword.

In the incompressible flow solver, the role of the outflow boundary conditions is to provide a computational boundary that is passive – particularly in the presence of strong vortical flow structures. Typically, this boundary condition is applied at boundaries that have been artificially imposed to emulate far-field conditions in a large physical domain.

For the **SET** option define the following card **Card Format**

Card 1	1	2	3	4	5	6	7	8
Variable	SSID							
Type	I							
Default	none							

For the **SEGMENT** option, define the following card.

Card Format

Card 1	1	2	3	4	5	6	7	8
Variable	N1	N2	N3	N4				
Type	I	I	I	I				
Default	none	none	none	none				

VARIABLE	DESCRIPTION
SSID	Segment set ID
N1, N2, ...	Node ID's defining the segment

9.1.2 *BOUNDARY_PRESCRIBED_CFD_*OPTION***

Available options include: **NODE** or **SET**

Purpose: Define an imposed nodal variable (velocity, temperature, species, etc.) on a node or a set of nodes for the Navier-Stokes flow solver. Do not use the NODE option in r-adaptive problems since the node ID's may change during the adaptive step.

The prescription of all nodal variables in the incompressible flow solver are defined using this keyword. It is similar in function to the *BOUNDARY_PRESCRIBED_MOTION_*OPTION* keyword but permits the specification of boundary conditions for velocities, mass concentration species, temperature, turbulent kinetic energy, etc. Additionally, the keyword optionally provides for prescribing all velocities for example on a no-slip and no-penetration surface. Multiple instances of the keyword permit individual nodal variables to be prescribed using independent load curves and scale factors.

Card Format

Card 1	1	2	3	4	5	6	7	8
Variable	*type*ID	DOF		LCID	SF			
Type	I	I		I	F			
Default	none	none		none	1.			

VARIABLE	DESCRIPTION
*type*ID	Node ID (NID) or node set ID (NSID), see *SET_NODE.
DOF	Applicable degrees-of-freedom EQ. 101: x-velocity, EQ. 102: y-velocity, EQ. 103: z-velocity, EQ. 104: temperature, EQ. 107: turbulent kinetic energy, EQ. 110: turbulent eddy viscosity, EQ. 121: species mass fraction-1, EQ. 122: species mass fraction-2, EQ. 123: species mass fraction-3, EQ. 124: species mass fraction-4, EQ. 125: species mass fraction-5, EQ. 126: species mass fraction-6, EQ. 127: species mass fraction-7, EQ. 128: species mass fraction-8, EQ. 129: species mass fraction-9, EQ. 130: species mass fraction-10, EQ. 201: x-, y-, z-velocity, EQ. 202: x-, y-velocity, EQ. 203: y-, z-velocity, EQ. 301: all species.
LCID	Load curve ID to describe motion value versus time, see *DEFINE_CURVE.
SF	Load curve scale factor (default=1.0).

Example Input:

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$$$$  *BOUNDARY_PRESCRIBED_CFD_SET
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$  A set of nodes is given a prescribed velocity in the
$  x-direction according to a specified vel-time curve (which is scaled).
$
*BOUNDARY_PRESCRIBED_CFD_SET
$
$...>....1....>....2....>....3....>....4....>....5....>....6....>....7....>....8
$     nsid       dof                lcid        sf
         4       101                   8       2.0
$
$     nsid = 4    nodal set ID number, requires a *SET_NODE_option
$      dof = 101  x-velocity is prescribed
$     lcid = 8    velocity follows load curve 8, requires a *DEFINE_CURVE
$       sf = 2.0  velocity specified by load curve is scaled by 2.0
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

9.1.3 *BOUNDARY_PRESSURE_CFD_SET

Purpose: Apply a pressure load over each segment in a segment set. The pressure convention follows Figure 9.1. This keyword activates what is commonly referred to as the pressure boundary condition according to §5.3.1.

Card Format

Card 1	1	2	3	4	5	6	7	8
Variable	SSID	LCID	P					
Type	I	I	F					
Default	none	none	none					
Remarks		1	1					

VARIABLE	DESCRIPTION
SSID	Segment set ID, see *SET_SEGMENT.
LCID	Load curve ID, see *DEFINE_CURVE.
P	Pressure to be applied.

Remarks:

1. The load curve multipliers may be used to increase or decrease the pressure. The time value is not scaled.

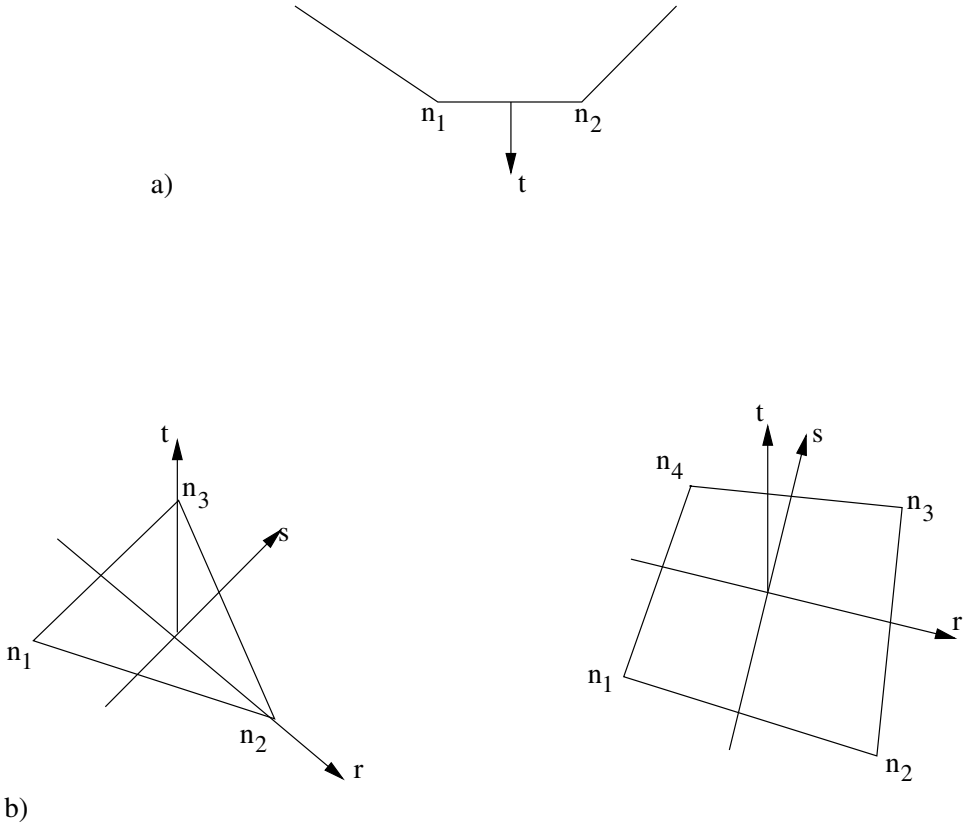


Figure 9.1: Node numbering for pressure boundary segments: a) two dimensional elements, b) three dimensional elements. Positive pressure acts in the negative t-direction. For two dimensional problems, repeat the second node for the third and fourth nodes in the segment definitions.

9.1.4 *CONTROL_CFD_AUTO

Purpose: Set the time-step control options for the Navier-Stokes flow solver. *CONTROL_CFD_GENERAL is used in conjunction with with keyword to control the flow solver time-integration options.

Card Format

Card 1	1	2	3	4	5	6	7	8
Variable	IAUTO	EPSDT	DTSF	DTMAX				
Type	I	F	F	F				
Default	0	1.0e-3	1.25	none				
Remarks	1			2				

VARIABLE	DESCRIPTION
IAUTO	Set the time step control type: EQ. 0: IAUTO=1 for fixed time-step size, EQ. 1: Fixed time step based on DTINIT (default) see *CONTROL_CFD_GENERAL EQ. 2: Time step based on CFL/stability for INSOL=3/1, EQ. 3: Automatic time-step selection based on local truncation error using a second-order Adams-Bashforth predictor and trapezoidal rule corrector.
EPSDT	Set the tolerance for the local truncation error in time. EQ. 0: EPSDT=1.0e-3 (default).
DTSF	Set the maximum time step scale factor that may be applied at any given time step. This sets the upper limit on the amount that the time step can be increased during any given step. EQ. 0: DTSF=1.25 (default).
DTMAX	Set the upper limit on the time step size. This value puts a ceiling on how far the time step may be increased for IAUTO=3. EQ. 0: 10.0*DTINT (default).

Remarks:

1. There are multiple solver options for a variety of flow-related physics in LS-DYNA. The selection of the time step control mechanism is dependent upon the flow solver that is selected. IAUTO=1 may be used with any of the solution methods. IAUTO=2 forces the time step to be based on either stability or the CFL number (see *CONTROL_CFD_GENERAL) for either the explicit (INSOL=1) or the semi-implicit (INSOL=3) methods. For IAUTO=2, the ICKDT parameter may be used to control the interval at which the time step is checked and adjusted. The use of the second-order predictor-corrector time step control is restricted to the fully-implicit solver, i.e., INSOL=3 and IADVEC=40.

2. For IAUTO=3, the default maximum time step, DTMAX, is set 10 times larger than the starting time step DTINIT.

9.1.5 *CONTROL_CFD_GENERAL

Purpose: Set the solver parameters for the Navier-Stokes flow solver. *CONTROL_CFD_MOMENTUM, *CONTROL_CFD_TRANSPORT, and *CONTROL_CFD_PRESSURE are used in conjunction with this keyword to control the flow solver options. Material models may be specified with the *MAT_CFD_*OPTION* keyword input and turbulence models are activated with the *CONTROL_CFD_TURBULENCE keyword input.

Card Format

Card 1	1	2	3	4	5	6	7	8
Variable	INSOL	DTINIT	CFL	ICKDT	IACURC			
Type	I	F	F	I	I			
Default	3	none	0.9 (2.0)	10	0			
Remarks	1				2			

VARIABLE	DESCRIPTION
INSOL	Set the solver type: EQ. 0: INSOL=3 (default), EQ. 1: Explicit, transient incompressible Navier-Stokes, EQ. 3: Semi-implicit/fully-implicit,transient, incompressible Navier-Stokes using staggered velocity-pressure.
DTINIT	Set the initial time step for the Navier-Stokes and all auxiliary transport equations. The time step is computed based on either the prescribed CFL number (INSOL=3) or stability (INSOL=1) unless ICKDT<0 or IAUTO=3 on the *CONTROL_CFD_AUTO keyword.
CFL	Set the maximum grid-CFL number to be maintained during the computation. EQ. 0: CFL=0.9 (default for INSOL=1), CFL=2 (default for INSOL=3)

VARIABLE	DESCRIPTION
ICKDT	Set the interval to check and report the grid Reynolds and advective CFL numbers. ICKDT<0 checks and reports the grid Reynolds and advective CFL numbers but does not modify the time step. ICKDT>0 modifies the time step according to the prescribed CFL limit and any required stability limits. The report of the grid Reynolds and CFL numbers to the screen may be toggled with the "grid" sense switch. EQ.0: ICKDT=10 (default).
IACURC	Activate the use of full-quadrature for certain terms in the momentum and transport equation. The accuracy flag improves the accuracy of body force calculations and certain advective/convective terms with a modest increase in computational time. EQ.0: don't use the increased quadrature rules (default), EQ.1: use increased quadrature on advective/convective/body force terms.

Remarks:

1. There are multiple solver options for a variety of flow-related physics in LS-DYNA. The selection of the incompressible/low-Mach flow physics and related flow solver is determined by the INSOL input on the *CONTROL_CFD_GENERAL keyword. Currently, there are two valid values for INSOL. INSOL=1 selects the explicit time integrator that requires the use of a lumped mass matrix. In this case, the IMASS, THETAK, THETAB, THETAA and THETAF variables associated with the *CONTROL_CFD_MOMENTUM keyword are ignored. INSOL=3 selects the semi-implicit projection algorithm which makes use of these variables.

2. This option is only available for the semi-implicit/implicit solution algorithm INSOL=3.

9.1.6 *CONTROL_CFD_MOMENTUM

Purpose: Set the solver parameters to be used for the momentum equations in the Navier-Stokes solver. Card 1 is used to control the time integrator and advective transport options. Card 2 is used to set the linear solver options such as the maximum iteration count and interval to check the convergence criteria.

Card Format

Card 1	1	2	3	4	5	6	7	8
Variable	IMASS	IADVEC	IFCT	DIVU	THETAK	THETAA	THETAF	
Type	I	I	I	F	F	F	F	
Default	1	10	1	1.0e-5	0.5	0.5	0.5	
Remarks	1	2	3	4	5	6	6	

Card 2	1	2	3	4	5	6	7	8
Variable	MSOL	MAXIT	ICHKIT	IWRT	IHIST	EPS	IHG	EHG
Type	I	I	I	I	I	F	I	F
Default	20	100	2	0	0	1.0e-5	1	1.0
Remarks	6							

VARIABLE	DESCRIPTION
IMASS	Select the mass matrix formulation to use: EQ.0: IMASS=1 (default), EQ.1: Lumped mass matrix, EQ.2: Consistent mass matrix, EQ.3: Higher-order mass matrix.

VARIABLE	DESCRIPTION
IADVEC	Toggle the treatment of advection between explicit with balancing tensor diffusivity or fully-implicit. EQ.0: IADVEC=10 for forward Euler with BTD (default), EQ.-1:IADVEC=0 for forward Euler without BTD, EQ.10: forward Euler with BTD, EQ.40: fully-implicit, simplified trapezoid rule.
IFCT	Toggle the use of the advective flux limiting advection scheme. EQ.0: IFCT=1 (default), EQ.1: Advective flux limiting is on, EQ.-1: Advective flux limiting is off.
DIVU	Set the RMS divergence tolerance, i.e., $\|\nabla \cdot \mathbf{u}\|_{RMS} \leq \varepsilon$. This tolerance is used for the initial startup procedure to insure that proper initial conditions are prescribed for the momentum equations. EQ.0: DIVU=1.0e-5 (default).
THETAK	Time weighting for viscous/diffusion terms. Valid values are $0 \leq \theta_K \leq 1$ with $\theta_K = 1/2$ for second-order accuracy in time. EQ.0: THETAK=0.5 (default).
THETAA	Time weighting for advection terms.
THETAF	Time weighting for body forces and boundary conditions. Valid values are $0 \leq \theta_F \leq 1$ with $\theta_F = 1/2$ for with for second-order accuracy in time. EQ.0: THETAF=0.5 (default).
MSOL	Set the equation solver type for the momentum equations. EQ.0: MSOL=20 (default), EQ.20: Jacobi preconditioned conjugate gradient method, EQ.30: Jacobi preconditioned conjugate gradient squared method. (default when IADVEC=40)
MAXIT	Set the maximum number of iterations for the iterative equation solver. EQ.0: MAXIT=100 (default).
ICHKIT	Set the interval to check the convergence criteria for the iterative equation solver. EQ.0: ICHKIT=2 (default).
IWRT	Activate the output of diagnostic information from the equation solver. EQ.0: Diagnostic information is off (default), EQ.1: Diagnostic information is on.

VARIABLE	DESCRIPTION
IHIST	Activate the generation of a convergence history file from the equation solver. The ASCII history files are "velx.his", "vely.his" and "velz.his" EQ.0: Convergence history is off (default), EQ.1: Convergence history is on.
EPS	Set the convergence criteria for the iterative equation solver. EQ.0: EPS=1.0e-5 (default).
IHG	Set the type of hourglass stabilization to be used with the momentum equations. This only applies to the explicit treatment of the momentum equations (INSOL=1). EQ.0: IHG=1 (default), EQ.1: LS-DYNA CFD viscous hourglass stabilization, EQ.2: γ-hourglass stabilization viscous form.
EHG	Set the hourglass stabilization multiplier. (see IHG above). EQ.0: EHG=1.0 (default).

Remarks:

1. The IMASS variable is only active when INSOL$\geq$2 on the *CONTROL_CFD_GENERAL keyword.
2. The balancing tensor diffusivity should always be used with the explicit forward-Euler treatment of the advection terms. This is the default.
3. The use of the flux limiting procedures is currently restricted to the explicit advective procedures.
4. DIVU sets the ceiling on the discrete divergence that is permitted during a simulation when INSOL=1. If the divergence at a given time step exceeds the value set by DIVU, then an intermediate projection is performed to return the velocity to a div-free state.
5. The time weighting variables only apply to the case when INSOL$\geq$2 on the *CONTROL_CFD_GENERAL keyword.
6. The MSOL keyword for the *CONTROL_CFD_MOMENTUM keyword only applies for INSOL$\geq$2 on the *CONTROL_CFD_GENERAL keyword.

9.1.7 *CONTROL_CFD_PRESSURE

Purpose: Set the pressure solver parameters to be used for the incompressible Navier-Stokes equations.

Card Format

Card 1	1	2	3	4	5	6	7	8
Variable	IPSOL	MAXIT	ICHKIT	IWRT	IHIST	EPS		
Type	I	I	I	I	I	F		
Default	22	200	5	0	0	1.0e-5		

Card 2	1	2	3	4	5	6	7	8
Variable	NVEC	ISTAB	BETA	SID	PLEV	LCID		
Type	I	I	F	I	F	I		
Default	5	1	0.05	none	0.0	0		

VARIABLE	DESCRIPTION
IPSOL	Set the pressure solver type: EQ.0: IPSOL=22 for serial, IPSOL=21 for MPP (default). EQ.10: Sparse direct solver, EQ.11: PVS direct solver, EQ.20: Jacobi preconditioned conjugate gradient method, EQ.21: SSOR preconditioned conjugate gradient method, EQ.22: SSOR preconditioned conjugate gradient using the Eisenstat transformation.

VARIABLE	DESCRIPTION
MAXIT	Set the maximum number of iterations for the pressure solver. EQ.0: MAXIT=200 (default).
ICHKIT	Set the interval to check the convergence criteria for the pressure solver. EQ.0: ICHKIT=5 (default).
IWRT	Activate the output of diagnostic information from the pressure solver. EQ.0: Diagnostic information is off (default), EQ.1: Diagnostic information is on. (NOTE: during execution, sense switch "lprint" can be used to toggle this flag on or off.)
IHIST	Activate the generation of a convergence history file for the pressure solver. The ASCII history file is "ppe.his". EQ.0: Convergence history is off (default), EQ.1: Convergence history is on.
EPS	Set the convergence criteria for the pressure solver. EQ.0: EPS=1.0e-5 (default).
NVEC	Set the number of A-conjugate vectors to use during the iterative pressure solve. EQ.0: NVEC=5 (default), LT.0: A-conjugate projection is disabled.
ISTAB	Set the stabilization type. EQ.0: ISTAB=2 (default), EQ.1: Local jump stabilization, EQ.2: Global jump stabilization, EQ.-1: No stabilization is active.
BETA	Stabilization parameter for ISTAB=1,2. Valid values for the stabilization parameter are $0 \leq \beta \leq 1$. EQ.0: BETA=0.05 (default).
SID	Solid element set or shell element set (see *SET_SOLID/SET_SHELL_OPTION) to be used for the prescription of hydrostatic pressure.
PLEV	Set the hydrostatic pressure level. This value multiplies the values of the load curve specified with the LCID option. EQ.0: PLEV=0.0 (default).
LCID	Load curve to be used for setting the hydrostatic pressure. By default, LCID=0 which forces a constant pressure level to be set at the level prescribed by PLEV. EQ.0: LCID=0 (default)

9.1.8 *CONTROL_CFD_TRANSPORT

Purpose: Activate the calculation of transport variables and associated solver parameters to be used for the auxiliary scalar transport equations. Card 1 is used to activate the auxiliary transport equations and Card 2 is used to set the mass matrix, advection, and time-weighting options. Card 3 is used to set the linear solver options such as the maximum iteration count and interval to check the convergence criteria.

Card Format

Card 1	1	2	3	4	5	6	7	8
Variable	ITEMP		NSPEC					
Type	I		I					
Default	0		0					
Remarks								

Card 2	1	2	3	4	5	6	7	8
Variable	IMASS	IADVEC	IFCT	THETAK	THETAA	THETAF		
Type	I	I	I	F	F	F		
Default	1	10	1	0.5	0.5	0.5		
Remarks	1	2	3	4	4	4		

Card 3	1	2	3	4	5	6	7	8
Variable	ITSOL	MAXIT	ICHKIT	IWRT	IHIST	EPS	IHG	EHG
Type	I	I	I	I	I	F	I	F
Default	20	100	2	0	0	1.0e-5	1	1.0
Remarks	5							

VARIABLE	DESCRIPTION
ITEMP	Solve the energy equation in terms of temperature. EQ.0: No energy equation (default), EQ.1: Energy equation is solved in terms of temperature.
NSPEC	Activate the solution of NSPEC species transport equations. EQ.0: No species equations are solved (default), EQ.NSPEC: Solve for NSPEC species. Up to 10 species transport equations may be active ($0 \leq NSPEC \leq 10$).
IMASS	Select the mass matrix formulation to use: EQ.0: IMASS=1 (default), EQ.1: Lumped mass matrix, EQ.2: Consistent mass matrix, EQ.3: Higher-order mass matrix.
IADVEC	Toggle the treatment of advection between explicit with balancing tensor diffusivity or fully-implicit. EQ.0: IADVEC=10 for forward Euler with BTD (default), EQ.-1:IADVEC=0 for forward Euler without BTD, EQ.10: forward Euler with BTD, EQ.40: fully-implicit with simplified trapezoid rule.
IFCT	Toggle the use of the advective flux limiting advection scheme. EQ.0: IFCT=1 (default), EQ.1: Advective flux limiting is on, EQ.-1: Advective flux limiting is off.

VARIABLE	DESCRIPTION
THETAK	Time weighting for the diffusion terms. Valid values are $0 \leq \theta_K \leq 1$ with $\theta_K = 1/2$ for second-order accuracy in time. EQ.0: THETAK=0.5 (default).
THETAA	Time weighting for advection terms.
THETAF	Time weighting for body forces and boundary conditions. Valid values are $0 \leq \theta_F \leq 1$ with $\theta_F = 1/2$ for with for second-order accuracy in time. EQ.0: THETAF=0.5 (default).
ITSOL	Set the equation solver type for the momentum equations. EQ.0: MSOL=20 (default), EQ.20: Jacobi preconditioned conjugate gradient method, EQ.30: Jacobi preconditioned conjugate gradient squared method. (default when IADVEC=40)
MAXIT	Set the maximum number of iterations for the iterative equation solver. EQ.0: MAXIT=100 (default).
ICHKIT	Set the interval to check the convergence criteria for the iterative equation solver. EQ.0: ICHKIT=2 (default).
IWRT	Activate the output of diagnostic information from the equation solver. EQ.0: Diagnostic information is off (default), EQ.1: Diagnostic information is on.
IHIST	Activate the generation of a convergence history file from the equation solver. EQ.0: Convergence history is off (default), EQ.1: Convergence history is on.
EPS	Set the convergence criteria for the iterative equation solver. EQ.0: EPS=1.0e-5 (default).
IHG	Set the type of hourglass stabilization to be used with the momentum equations. This only applies to the explicit treatment of the momentum equations (INSOL=1). EQ.0: IHG=1 (default), EQ.1: LS-DYNA CFD viscous hourglass stabilization, EQ.2: γ-hourglass stabilization viscous form.
EHG	Set the hourglass stabilization multiplier. (see IHG above). EQ.0: EHG=1.0 (default).

Remarks:

1. The IMASS variable is only active when INSOL≥2 on the *CONTROL_CFD_GENERAL keyword.
2. The balancing tensor diffusivity should always be used with the explicit treatment of the advection terms. (This is the default.)
3. The use of the flux limiting procedures is currently restricted to the explicit advective procedures.
4. The time weighting variables only apply to the case when INSOL≥2 on the *CONTROL_CFD_GENERAL keyword.
5. The ITSOL keyword for the *CONTROL_CFD_TRANSPORT keyword only applies for INSOL≥2 on the *CONTROL_CFD_GENERAL keyword.

9.1.9 *CONTROL_CFD_TURBULENCE

Purpose: Activate a turbulence model and set the associated model parameters.

Card Format

Card 1	1	2	3	4	5	6	7	8
Variable	ITRB	C1	C2	C3	C4	C5	C6	C7
Type	I	F	F	F	F	F	F	F
Default	0	-	-	-	-	-	-	-
Remarks		1	1	1	1	1	1	1

VARIABLE	DESCRIPTION
ITRB	Select the turbulence model: EQ.0 : Turbulence models are disabled (default), EQ.1 : Smagorinsky LES subgrid scale model,
C1 ... C7	Turbulence model constants. See the table below for the model-specific definition of the constants and their default values.

Remarks:

1. The number and value of the input constants are model-specific. See the table below for definition of the constants and their default values.

ITRB	Turbulence Model	C1	C2	C3	C4	C5	C6	C7
1	Smagorinsky LES SGS	$C_S = 0.1$						

9.1.10 *INITIAL_CFD

Purpose: Specify initial conditions for all nodal variables.

Card Format (1 of 3)

Card 1	1	2	3	4	5	6	7	8
Variable	U	V	W	T	H	RHO	Z1	Z2
Type	F	F	F	F	F	F	F	F
Default	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Card Format (2 of 3)

Card 2	1	2	3	4	5	6	7	8
Variable	Z3	Z4	Z5	Z6	Z7	Z8	Z9	Z10
Type	F	F	F	F	F	F	F	F
Default	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Card Format (3 of 3)

Card 3	1	2	3	4	5	6	7	8
Variable	K	EPS						
Type	F	F						
Default	0.0	0.0						

VARIABLE	DESCRIPTION
U	Initial x-velocity
V	Initial y-velocity
W	Initial z-velocity
T	Initial temperature
H	Initial enthalpy
RHO	Initial density
Z1	Initial Species-1 concentration
Z2	Initial Species-2 concentration
...	
Z10	Initial Species-10 concentration
K	Intital turbulent kinetic energy
EPS	Initial turbulent dissipation rate

9.1.11 *MAT_CFD_*OPTION*

The ***MAT_CFD_** cards allow fluid properties to be defined in a stand-alone fluid analysis or in a coupled fluid/structure analysis; see ***CONTROL_SOLUTION** in the LS-DYNA Keyword User's Manual.

The only *OPTION* currently available is **CONSTANT**.

This is material property type 150. It allows constant, isotropic fluid properties to be defined for the incompressible/low-Mach CFD solver.

Card Format (1 of 3)

Card 1	1	2	3	4	5	6	7	8
Variable	MID	RHO	MU	K	CP	BETA	TREF	
Type	I	F	F	F	F	F	F	
Default	-	-	-	-	-	-	-	

Card Format (2 of 3)

Card 2	1	2	3	4	5	6	7	8
Variable	GX	GY	GZ	DIFF1	DIFF2	DIFF3	DIFF4	DIFF5
Type	F	F	F	F	F	F	F	F
Default	-	-	-	-	-	-	-	-

Card Format (3 of 3)

Card 3	1	2	3	4	5	6	7	8
Variable	DIFF6	DIFF7	DIFF8	DIFF9	DIFF10			
Type	F	F	F	F	F			
Default	-	-	-	-	-			

VARIABLE	DESCRIPTION
MID	Material identification, a unique number has to be chosen.
RHO	Fluid density
MU	Fluid viscosity
K	Thermal Conductivity
CP	Heat capacity
BETA	Coefficient of expansion
TREF	Reference temperature
GX	Gravitational acceleration in x-direction
GY	Gravitational acceleration in y-direction
GZ	Gravitational acceleration in z-direction
DIFF1	Species-1 diffusivity
DIFF2	Species-2 diffusivity
DIFF3	Species-3 diffusivity
...	
DIFF10	Species-10 diffusivity

9.2 Shared Keywords

The incompressible flow solver is an integral part of LS-DYNA, and because of this it shares keywords with the other physics options available in LS-DYNA. This section identifies the shared keywords and defines their interaction with the flow solver.

***CONTROL_SOLUTION.** Specify the analysis solution procedure to be used. Here, *SOLN=4* activates the incompressible flow solver in a stand-alone mode.

***CONTROL_OUTPUT.** Set miscellaneous output options. For the flow solver, *NPOPT=0* echos flow velocities and pressures to the *d3hsp* file.

***CONTROL_TERMINATION.** Stop the job. The *EDENG* and *ENDMAS* parameters are reserved for structural analyses.

***DATABASE_BINARY_D3PLOT.** Set the time interval for the output of state databases for plotting purposes. For incompressible flow calculations, LS-DYNA generates an augmented state database that contains flow-specific variables, e.g., vorticity, stream function, etc.

***DATABASE_BINARY_D3THDT.** Set the time interval for the output of time-history data. For incompressible flow calculations, LS-DYNA generates a separate binary time-history database that contains flow-specific variables, e.g., vorticity, stream function, etc. This new database is the *d3thins* database.

***DATABASE_BINARY_D3MEAN.** Define the starting time, the time window (in time steps), and the level of statistics to be used in creating a time-averaged database of flow statistics.

***DATABASE_BINARY_D3DUMP.** Define the output frequency in cycles for the binary restart files.

***DATABASE_BINARY_RUNRSF.** Define the output frequency in cycles for the running binary restart files.

***DATABASE_HISTORY_NODE.** Control which nodes are output to the binary database *D3THDT* and the ASCII file *NODOUT*.

***DEFINE_CURVE.** Define a load curve to be used for the prescription of time-dependent boundary conditions.

***SECTION_SHELL.** Define the section properties for 2-D (shell) elements. This keyword is used for 2-D Cartesian flow problems to select the element formulation used with the Navier-Stokes solver.

***SECTION_SOLID.** Define the section properties for the 3-D elements. This keyword is used for 3-D flow problems to select the element formulation used with the Navier-Stokes solver.

Chapter 10

Example Problems

This chapter presents several LS-DYNA flow calculations for the first time user who wishes to perform verification computations for comparison before embarking on a full-blown analysis. For this reason, the control files are replicated here with representative results that can be used to check the local LS-DYNA installation. For the most part, sample problems presented here use relatively coarse meshes to minimize run times and provide a starting point for the user who wishes to experiment with code options before attempting any significant calculations. The example problems in this chapter may be obtained by contacting Livermore Software Technology Corporation.

All of the example problems presented in this chapter use the following nomenclature. The example input files are prefixed by the problem name, e.g., `cylinder`. File names that contain `_dat.k` as a suffix contain the nodal coordinates, element connectivity and all sets for the problem. The file names with a `_p2.k` or `_fe.k` suffix contain the primary control keywords for the semi-/fully-implicit projection and explicit solution methods respectively. For example, `cylinder_p2.k` will contain the primary control keywords for the `cylinder` flow problem and will `*INCLUDE` the `cylinder_dat.k` data file. Details on the `*INCLUDE` keyword may be found in the LS-DYNA Keyword User's Manual [27].

10.1 Poiseuille Flow

Poiseuille flow in a channel is characterized by a balance between a constant pressure difference and viscous shear forces, i.e.,

$$\frac{1}{Re}\frac{\partial^2 u}{\partial y^2} = \frac{\partial p}{\partial x}, \tag{10.1}$$

where Re is the Reynolds number based on channel height H, and u, p, and x, y are the non-dimensional x-velocity, pressure and coordinates. The 2-D channel flow problem considered here consists of a 2-D duct with a 5:1 aspect ratio, $Re = 100$, and $\partial p/\partial x = -0.12$. This choice of Reynolds number and pressure gradient results in a steady parabolic velocity profile with $u_{max} = 1.5$ and $u_{avg} = 1.0$. Here, the non-dimensional equations were obtained using a length scale H, velocity scale U, and time scale U/H. Figure 10.1 shows the mesh with node and segment set id's for the entrance region.

In order to represent the constant pressure gradient, an inflow pressure boundary condition of $p = 0.6$ is prescribed with $v = 0$. At the inflow boundary, the pressure boundary condition is prescribed using the

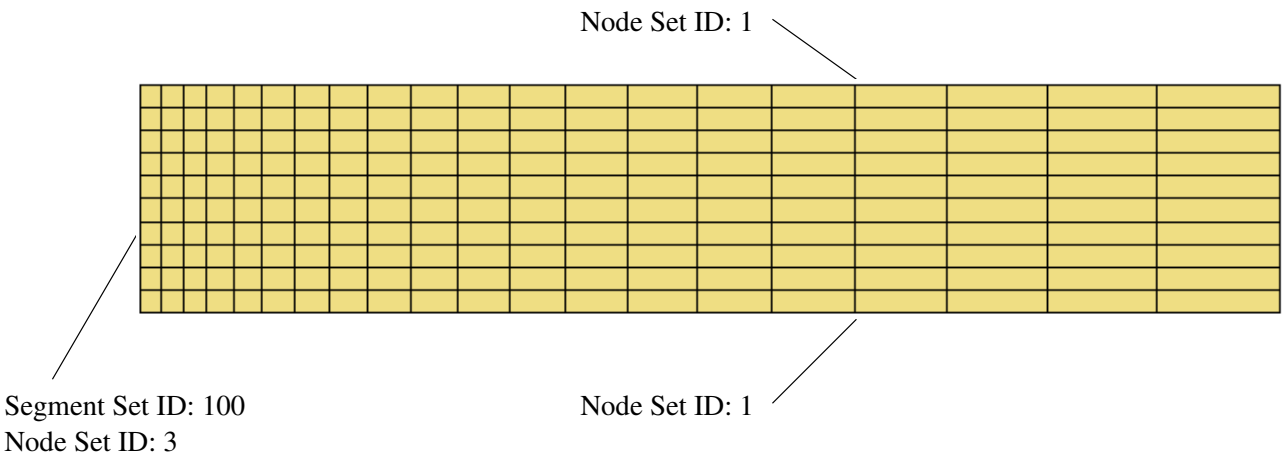


Figure 10.1: Mesh for 2-D Entrance Region.

*BOUNDARY_PRESSURE_CFD_SET keyword with segment set id 100. Homogeneous outflow boundary conditions are used at the outflow resulting in $p \approx 0$. At the top and bottom wall, no-slip and no-penetration boundary conditions ($u = v = 0$) are applied using node set id 1.

There are multiple flow solution algorithms available in LS-DYNA that may be applied to this problem: 1) an explicit time integration method with a reduced-integration element, 2) a semi-implicit projection method with explicit advection, and 3) the fully-implicit projection method with a "smart" time integrator. In this example, a the semi-implicit projection method is applied with the selection of a backward-Euler treatment of the viscous terms and a row-sum lumped mass matrix.

The LS-DYNA keyword file for this problem is shown below in a `typewriter` font. The keyword input is segregated into 5 primary parts: 1) the control keywords, 2) load curves, 3) boundary conditions, 4) fluid properties, and 5) I/O controls. In this example, the second-order projection algorithm (`insol=3` on the *CONTROL_CFD_GENERAL keyword) is used with the viscous terms treated in a fully implicit mode, a lumped mass matrix, and explicit advection terms. This is obtained by setting `thetak=1.0`, `imass=1` and `iadvec=10` on the *CONTROL_CFD_MOMENTUM keyword. The *CONTROL_CFD_AUTO keyword is used to enable automatic time-step selection based on the maximum CFL number specified on the *CONTROL_CFD_GENERAL keyword, i.e., `cfl=2.0`. In addition, the initial time step is set to `dtinit=0.01`. A direct solver `ipsol=11` via the *CONTROL_CFD_PRESSURE keyword is used with a termination of `endcyc=2000` time steps. Constant fluid properties are set using the *MAT_CFD_CONSTANT keyword which, in the non-dimensional context, requires a unit density with viscosity $\mu = 1/Re$. The *DATABASE_BINARY_*OPTION* keywords are used to write a graphics state to the *d3plot* every 10 time units, nodal time history every time step, and a running restart every 1000 time steps. The input files, `duct_dat.k`, `duct_p2.k` and `duct_fe.k`, may be found in the `duct` sub-directory of the 2-D examples.

Example: `duct_p2.k`

```
*KEYWORD
*TITLE
Simple duct - entrance region
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
```

```
$ Setup the CFD problem - soln=4
$
*CONTROL_SOLUTION
$     soln
         4
*CONTROL_CFD_AUTO
$    iauto     epsdt      dtsf     dtmax
         2
*CONTROL_CFD_GENERAL
$    insol    dtinit       cfl     ickdt    istats    tstart      iavg
         3     0.010       2.0       100
*CONTROL_CFD_MOMENTUM
$    imass    iadvec      ifct      divu    thetak    thetaa    thetaf
         1        10                           1.0
$     msol     maxit    ichkit      iwrt     ihist       eps       ihg       ehg
        20       100         2         0         0
*CONTROL_CFD_PRESSURE
$    ipsol     maxit     ickit      iwrt     ihist       eps
        11
$     nvec      stab      beta       sid      plev      lcid

*CONTROL_TERMINATION
$   endtim    endcyc     dtmin     endeng   endmas
1000000.00      2000
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ Define the load curves
$
*DEFINE_CURVE
$     lcid      sidr       sfa       sfo      offa      offo    dattyp
         1
$                a1                  o1
                0.0                 1.0
          1000000.0                 1.0
*DEFINE_CURVE
$     lcid      sidr       sfa       sfo      offa      offo    dattyp
         2
$                a1                  o1
                0.0                 0.0
          1000000.0                 0.0
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ Setup the boundary conditions
$
$ Pressure boundary conditions
$
*BOUNDARY_PRESSURE_CFD_SET
$...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|....8
$     ssid      lcid         p
       100         1       0.6
```

```
$
$ Prescribed inlet BC
$
*BOUNDARY_PRESCRIBED_CFD_SET
$     nsid       dof               lcid        sf
         3       102                  2 1.0000000
$
$ Setup the no-slip BC's
$
*BOUNDARY_PRESCRIBED_CFD_SET
$     nsid       dof               lcid        sf
         1       101                  2 1.0000000
*BOUNDARY_PRESCRIBED_CFD_SET
$     nsid       dof               lcid        sf
         1       102                  2 1.0000000
*BOUNDARY_PRESCRIBED_CFD_SET
$     nsid       dof               lcid        sf
         2       101                  2 1.0000000
*BOUNDARY_PRESCRIBED_CFD_SET
$     nsid       dof               lcid        sf
         2       102                  2 1.0000000
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ Setup fluid properties
$
*MAT_CFD_CONSTANT
$      mid        ro        mu         k        Cp      beta      Tref
         1    1.0000    1.0e-2
$       gx        gy        gz     diff1     diff2     diff3     diff4     diff5

$    diff6     diff7     diff8     diff9    diff10

$
$ Section & part
$
*SECTION_SHELL
$      sid    elform      shrf       nip     propt   qr/irid     icomp
         1        31
$       t1        t2        t3        t4      nloc
 1.0000000 1.0000000 1.0000000 1.0000000
*PART
Fluid domain
$      pid     secid       mid     eosid      hgid      grav    adpopt      tmid
         1         1         1         0         0         0         0         0
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ Handle output of state, time-history,  and restart data
$
*DATABASE_BINARY_D3PLOT
$  dt/cycl      lcdt      beam     npltc
```

```
 10.0                          0
*DATABASE_BINARY_D3THDT
$  dt/cycl      lcdt      beam     npltc
 1.0e-6
*DATABASE_HISTORY_NODE
$      nd1       nd2       nd3       nd4       nd5       nd6       nd7       nd8
         6        28        39        50       215       226
*DATABASE_BINARY_RUNRSF
$  dt/cycl      lcdt      beam     npltc
      1000
*DATABASE_BINARY_D3DUMP
$  dt/cycl      lcdt      beam     npltc
         0
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
*INCLUDE
duct_dat.k
*END
```

Example: `duct_p2.k`

The grid CFL and Reynolds numbers were reported every 100 time steps by setting `ickdt=100` on the *CONTROL_CFD_GENERAL keyword. Figure 10.2 shows a sample of the screen output with the CFL limited time step and associated grid parameters. When the grid parameters are reported, they are presented in terms of the `xi-grid`, `eta-grid` (and `zeta-grid`) directions which correspond to the element-local ξ, η, and ζ coordinate directions as described in Chapter 3. For each local coordinate direction, a minimum and maximum grid CFL and Reynolds number is reported along with the corresponding element id. In this example, the time step is controlled so that the maximum grid CFL number never exceeds 2.0. During the calculation, the *lprint* sense switch toggles the output of the minimum/maximum pressure and velocities with the RMS divergence at each time step. Sample screen output of the reported minimum/maximum pressure and velocities with the RMS divergence is shown in Figure 10.3.

The fully-implicit treatment of the viscous terms is based on a first-order Forward-Euler time-integrator that in combination with a lumped mass matrix yields a rudimentary marching scheme for obtaining a steady-state solution to the channel flow problem. In order to verify that a steady-state has been achieved, nodal time-histories of the x-velocity along the centerline of the channel at the inlet and outlet boundaries are shown in Figure 10.4. In addition, a time-history of the global kinetic energy ($1/2\mathbf{u}^T M_L \mathbf{u}$) is shown in Figure 10.5. As shown by these plots, a steady-state solution is achieved after approximately 100 time units.

A plot of the velocity profile and the corresponding linear pressure field are presented in Figures 10.6 and 10.7. In Figure 10.6, velocity vectors are plotted at the outlet boundary to show the parabolic profile. The outlet velocity profile is compared to the exact solution to the Poiseuille problem in Figure 10.8 where it is seen that the computed velocities interpolate the exact solution at the nodal points.

```
g r i d  p a r a m e t e r s

   CFL based time step:
   ====================
   Local coordinate direction.....    xi
   Element number.................    101
   Element length................. 8.7298E-02
   Grid Reynolds no............... 9.2771E-01
   Grid CFL number................ 2.0000E+00
   CFL based time step............ 1.1877E-01

   Maximum xi-grid Reynolds numbers:
   =================================
   Element no.....................    200
   Element length................. 5.3391E-01
   Grid Reynolds number........... 1.0060E+01

   Maximum eta-grid Reynolds numbers:
   ==================================
   Element no..................... 97
   Element length................. 1.0000E-01
   Grid Reynolds number........... 3.5299E-05

   Maximum xi-grid CFL number:
   ===========================
   Element no.....................    101
   Element length................. 8.7298E-02
   Grid CFL number................ 2.0000E+00

   Maximum eta-grid CFL number
   ===========================
   Element no..................... 97
   Element length................. 1.0000E-01
   Grid CFl number................ 4.1926E-06
```

Figure 10.2: Screen report of grid parameters showing the minimum and maximum grid Reynolds and CFL numbers and and associated element numbers.

```
initialization completed

       0 t 0.0000E+00 div(u)= 0.0000E+00 pmn/pmx=0.320E-01/0.595E+00
        (u_i)mn/mx=0.000E+00/0.000E+00 0.000E+00/0.000E+00
       0 t 0.0000E+00 dt 1.00E-02 write d3plot file
       1 t 1.0000E-02 div(u)= 8.4941E-21 pmn/pmx=0.320E-01/0.595E+00
        (u_i)mn/mx=0.000E+00/0.120E-02 -.481E-09/0.426E-09
       2 t 2.0000E-02 div(u)= 1.6672E-20 pmn/pmx=0.320E-01/0.595E+00
        (u_i)mn/mx=0.000E+00/0.240E-02 -.954E-09/0.848E-09
       3 t 3.0000E-02 div(u)= 2.9852E-20 pmn/pmx=0.320E-01/0.595E+00
        (u_i)mn/mx=0.000E+00/0.360E-02 -.142E-08/0.127E-08
       4 t 4.0000E-02 div(u)= 4.5121E-20 pmn/pmx=0.320E-01/0.595E+00
        (u_i)mn/mx=0.000E+00/0.480E-02 -.188E-08/0.168E-08
       5 t 5.0000E-02 div(u)= 6.5703E-20 pmn/pmx=0.320E-01/0.595E+00
        (u_i)mn/mx=0.000E+00/0.600E-02 -.234E-08/0.209E-08

       .
       .
       .

    1995 t 2.0907E+02 div(u)= 3.7079E-16 pmn/pmx=0.320E-01/0.595E+00
         (u_i)mn/mx=0.000E+00/0.150E+01 -.261E-07/0.294E-07
    1996 t 2.0919E+02 div(u)= 3.7104E-16 pmn/pmx=0.320E-01/0.595E+00
         (u_i)mn/mx=0.000E+00/0.150E+01 -.261E-07/0.294E-07
    1997 t 2.0931E+02 div(u)= 3.7186E-16 pmn/pmx=0.320E-01/0.595E+00
         (u_i)mn/mx=0.000E+00/0.150E+01 -.261E-07/0.294E-07
    1998 t 2.0942E+02 div(u)= 3.7329E-16 pmn/pmx=0.320E-01/0.595E+00
         (u_i)mn/mx=0.000E+00/0.150E+01 -.261E-07/0.294E-07
    1999 t 2.0954E+02 div(u)= 3.7428E-16 pmn/pmx=0.320E-01/0.595E+00
         (u_i)mn/mx=0.000E+00/0.150E+01 -.261E-07/0.294E-07
    2000 t 2.0966E+02 div(u)= 3.7498E-16 pmn/pmx=0.320E-01/0.595E+00
         (u_i)mn/mx=0.000E+00/0.150E+01 -.261E-07/0.294E-07
    2000 t 2.0966E+02 dt 1.19E-01 write d3plot file
    2000 t 2.0966E+02 dt 1.19E-01 write runrsf file
```

Figure 10.3: Screen output of RMS divergence, minimum/maximum pressure and velocity values.

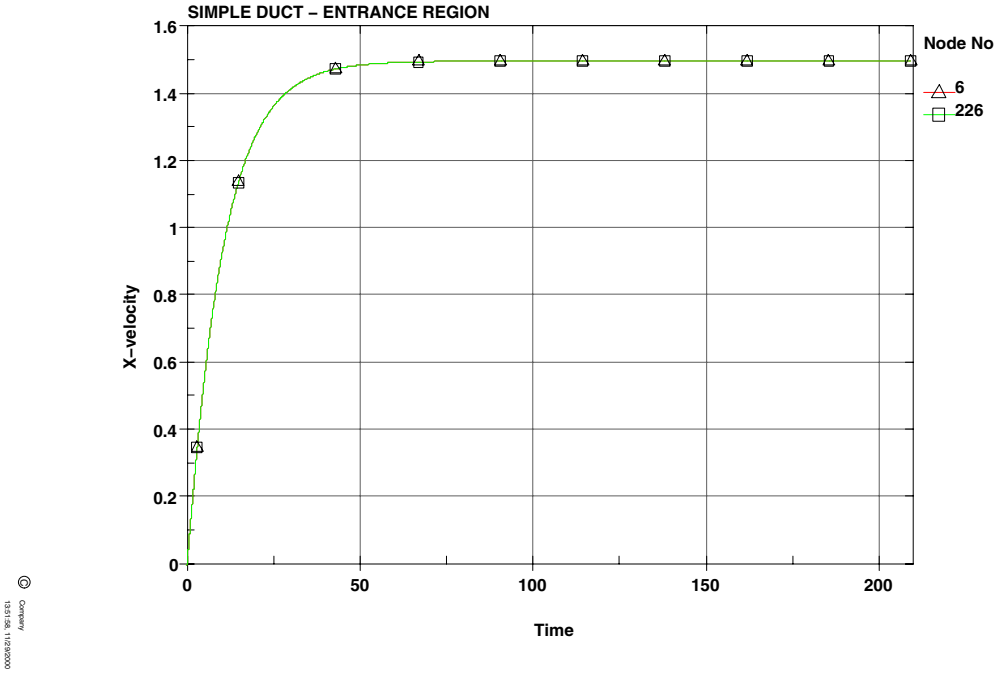


Figure 10.4: Nodal time history plot for the x-velocity along the centerline at the inlet (node 5) and the outlet (node 226).

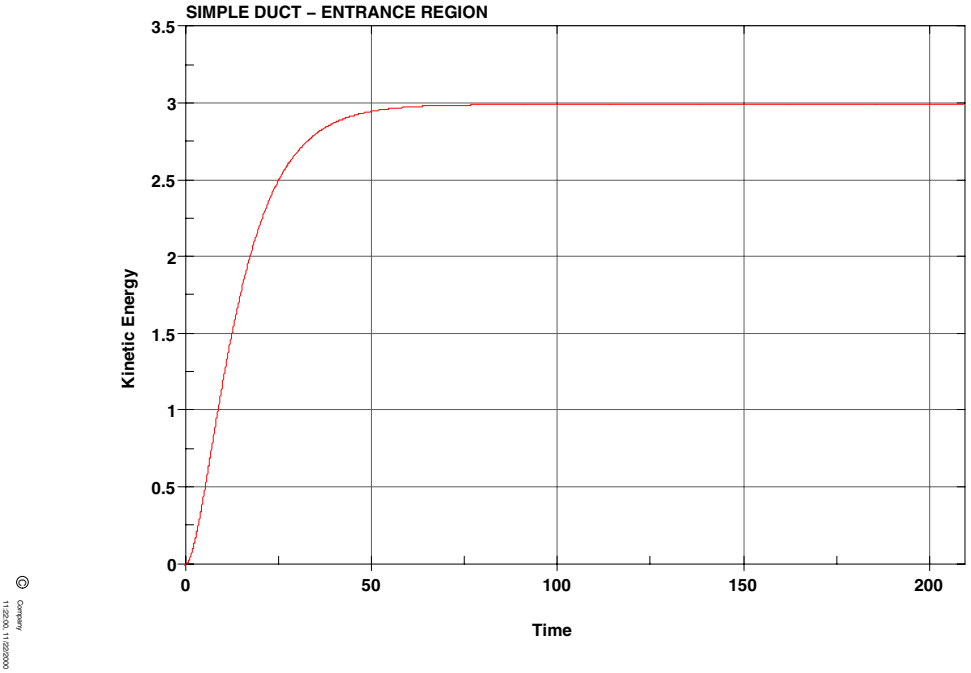


Figure 10.5: Kinetic energy time history.

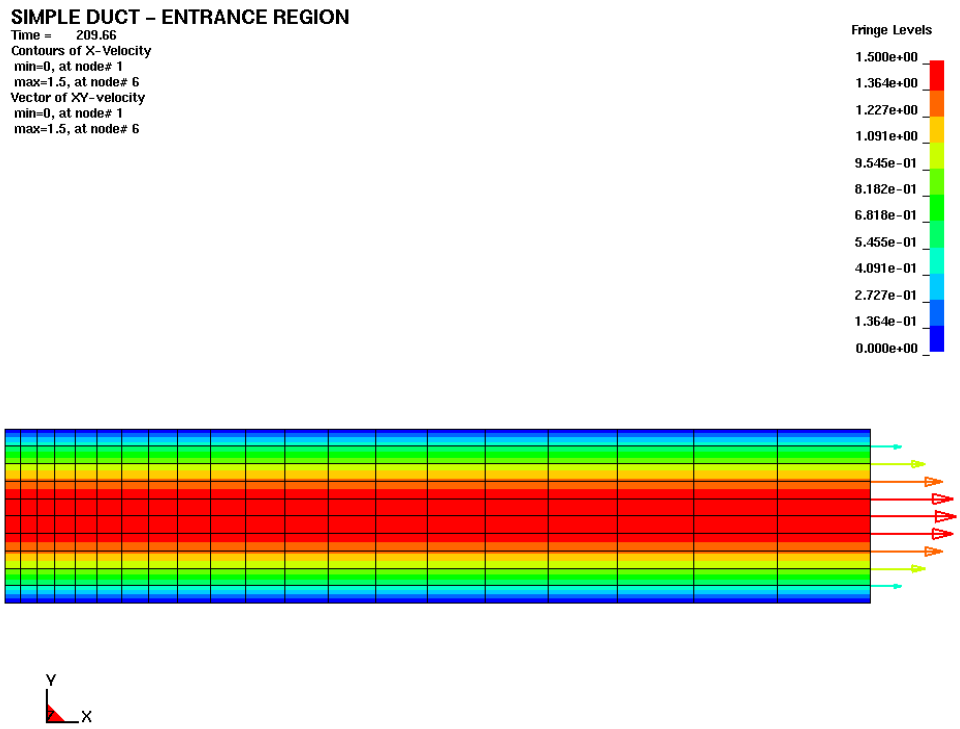


Figure 10.6: X-velocity field after 209.66 time units.

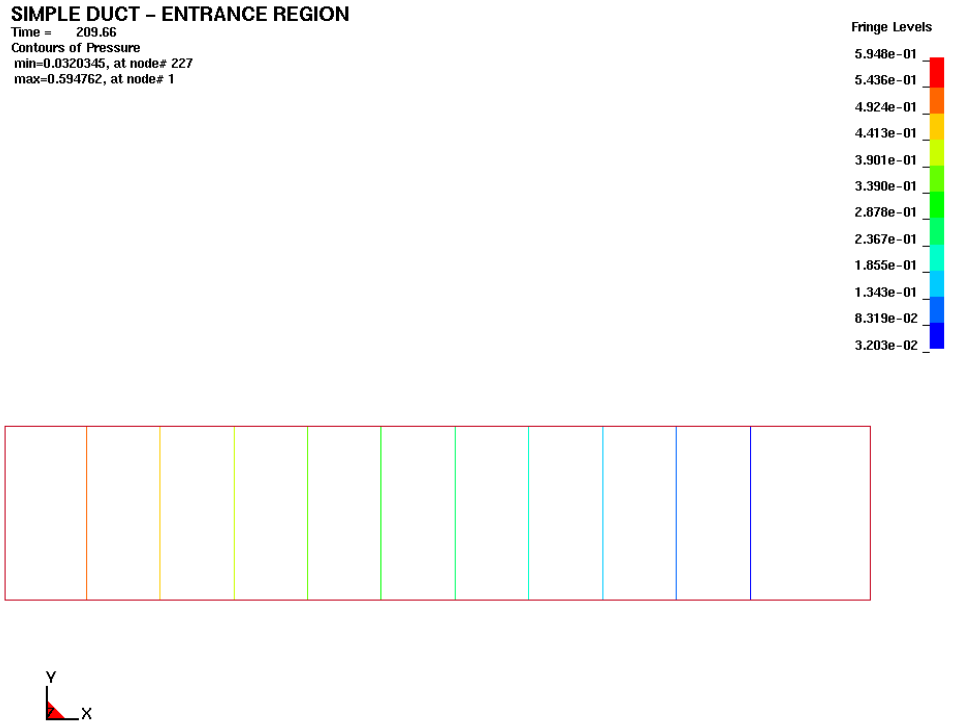


Figure 10.7: Pressure contours after 209.66 time units.

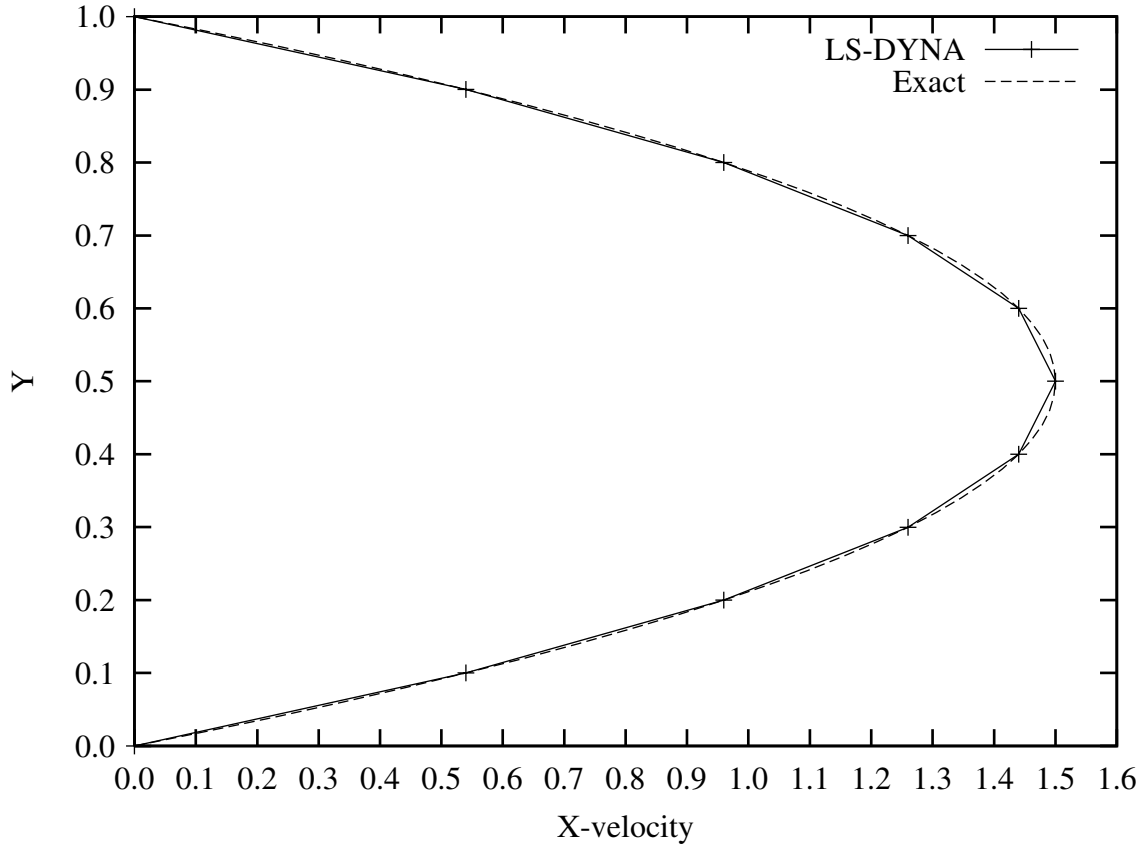


Figure 10.8: Steady-state x-velocity profile at outlet plane.

10.2 Natural Convection in a Square Cavity

The thermal cavity benchmark introduced by De Vahl Davis [99, 100] is used here to demonstrate the application of the explicit, semi-implicit and fully-implicit algorithms to buoyancy-driven flow. Figure 10.9 shows the computational domain for the differentially heated cavity. The non-dimensional governing equations for the time-dependent thermal convection problem (in vector form) are the incompressible Navier-Stokes equations, conservation of mass, and the energy equation written in terms of temperature:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla P + Pr \nabla^2 \mathbf{u} + RaPr \hat{j} \theta, \tag{10.2}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{10.3}$$

and

$$\frac{\partial \theta}{\partial t} + \mathbf{u} \cdot \nabla \theta = \nabla^2 \theta, \tag{10.4}$$

where $\mathbf{u} = (u, v)$, P and θ are the velocity, the deviation from hydrostatic pressure, and temperature respectively, and $\hat{j}$ the unit vector in the y-direction. These non-dimensional equations were obtained using the characteristic length L, velocity $U = \alpha/L$, time scale $\tau = \alpha/L^2$, and pressure $\tilde{P} = \rho U^2$. Here, ρ is the mass density, g the gravitational acceleration, $\alpha = k/\rho C_p$ is the thermal diffusivity, and ν is the kinematic viscosity. The Prandtl number is $Pr = \nu/\alpha$ and fixed at $Pr = 0.71$, and the Rayleigh number is

$$Ra = \frac{g\beta(T_h - T_c)L^3}{\nu\alpha}, \tag{10.5}$$

where $T_h - T_c$ the temperature difference between the hot and cold walls, and β the coefficient of thermal expansion. The non-dimensional temperature is defined in terms of the wall temperature difference and a reference temperature as

$$\theta = \frac{T - T_c}{T_h - T_c}, \tag{10.6}$$

where T_h is the prescribed temperature of the hot wall, and T_c is the temperature of the cold wall.

The boundary conditions for this problem consist of no-slip and no-penetration walls with the top and bottom walls insulated. The initial conditions are prescribed with $u = v = 0$ and $T = (T_h + T_c)/2$. The LS-DYNA control file for the mesh with 441 nodes is shown below for the semi-implicit projection method (`box_p2.k`).

Similar to the `duct_p2.k` example, the `box_p2.k`, uses a fixed $CFL = 2$ condition with the automatic time-step selection and an initial time step of `dtinit=1.0e-4`. In order to solve an energy equation in conjunction with the momentum equations, `itemp=1` on the *CONTROL_CFD_TRANSPORT keyword. In addition, boundary conditions specifying $\theta = 1$ on the left wall (node set 2) and $\theta = 0$ on the right wall (node set 3) are included in the keyword input. In this example, the fluid properties have been translated into non-dimensional parameters, i.e., $\mu = Pr$, $\beta = RaPr$ with the inclusion of the gravity vector $g_y = -1$ and a reference temperature $\theta_{ref} = 0.5$.

As a basis for comparison, the explicit, semi-implicit and fully-implicit methods have been applied to this problem. For the explicit method, a fixed time-step was used with $\Delta t = 2.0e-4$. This requires `iauto=1` on the *CONTROL_CFD_AUTO keyword. The semi-implicit method started with a time-step $\Delta t = 1.0e-4$ and increased the time-step according to the maximum $CFL = 0.5$ condition (`iauto=2` on the *CONTROL_CFD_AUTO keyword). The fully-implicit method used a second-order Adams-Bashforth

predictor with a trapezoid rule corrector with the local time-truncation error specified as `epsdt=0.001` and `iauto=3` on the *CONTROL_CFD_AUTO keyword. Both the semi-implicit and fully-implicit methods used consistent mass matrices `imass=2` for the momentum and energy equations in conjunction with the default second-order time-weighting parameters for the viscous/diffusion and advective terms.

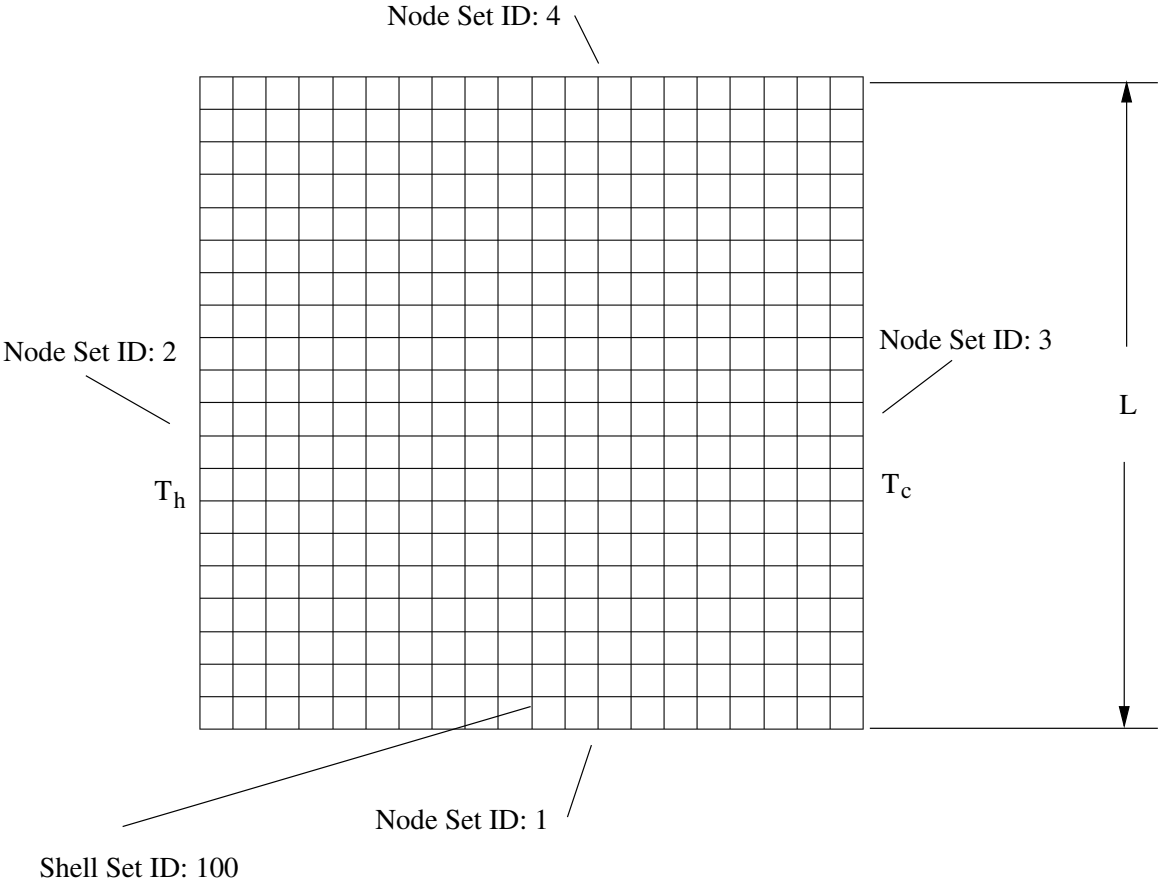


Figure 10.9: 21 x 21 thermal cavity mesh.

Example: box_p2.k

```
*KEYWORD
*TITLE
Thermal Cavity - Ra=1000
$
$$$
```

```
$     lcid      sidr       sfa       sfo      offa      offo    dattyp
         2
$                  a1                  o1
                  0.0                 0.0
            1000000.0                 0.0
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ Setup the boundary conditions
$
$ Setup the no-slip BC's -- set all velocity components at once
$
*BOUNDARY_PRESCRIBED_CFD_SET
$     nsid       dof                lcid        sf
         1       202                   2 1.0000000
*BOUNDARY_PRESCRIBED_CFD_SET
$     nsid       dof                lcid        sf
         2       202                   2 1.0000000
*BOUNDARY_PRESCRIBED_CFD_SET
$     nsid       dof                lcid        sf
         3       202                   2 1.0000000
*BOUNDARY_PRESCRIBED_CFD_SET
$     nsid       dof                lcid        sf
         4       202                   2 1.0000000
$
$ Temperature BC's
$
*BOUNDARY_PRESCRIBED_CFD_SET
$     nsid       dof                lcid        sf
         2       104                   1 1.0000000
*BOUNDARY_PRESCRIBED_CFD_SET
$     nsid       dof                lcid        sf
         3       104                   2 1.0000000
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ Setup non-dimensional Boussinesq fluid properties
$
*MAT_CFD_CONSTANT
$      mid        ro        mu         k        Cp      beta      Tref
         1    1.0000      0.71       1.0       1.0     710.0       0.5
$       gx        gy        gz     diff1     diff2     diff3     diff4     diff5
        0.0      -1.0
$    diff6     diff7     diff8     diff9    diff10


$
$ Section & part
$
*SECTION_SHELL
$      sid    elform      shrf       nip     propt   qr/irid     icomp
         1        31
$       t1        t2        t3        t4      nloc
```

```
 1.0000000 1.0000000 1.0000000 1.0000000
*PART
Fluid domain
$      pid     secid       mid     eosid      hgid      grav    adpopt      tmid
         1         1         1         0         0         0         0         0
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ Handle output of state, time-history,  and restart data
$
*DATABASE_BINARY_D3PLOT
$  dt/cycl      lcdt      beam     npltc
 1.0
*DATABASE_BINARY_D3THDT
$  dt/cycl      lcdt      beam     npltc
 1.0e-6
*DATABASE_HISTORY_NODE
$      nd1       nd2       nd3       nd4       nd5       nd6       nd7       nd8
        49       115       291       162       392       167       227       397
*DATABASE_BINARY_RUNRSF
$  dt/cycl      lcdt      beam     npltc
      1000
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
*INCLUDE
box_dat.k
*END
```

Example: `box_p2.k`

Sample screen output from the fully-implicit computation is shown in Figure 10.10. This output reports the minimum/maximum pressure, velocities and temperature as well as the current local time-truncation error `LTE`, new and old time steps, and the limiting minimum/maximum nodal differences that the local time-truncation error is based on.

Although time-accurate methods were used for this computation, a steady-state solution is obtained in each case. The time-history plot of the kinetic energy verifies this as shown in Figure 10.11 The minimum and maximum time-steps used by each method over the computation are summarized with the peak velocities in Table 10.1. The maximum time-step for the fully-implicit method was limited to 1.76e-3 while the semi-implicit method found a maximum time-step nearly 4 times larger. This is due to the fact that the local time-truncation error was limited to 1.0e-3 for the fully-implicit method, while the time-step calculation for the semi-implicit method was based solely on the convective stability. That is to say, the semi-implicit method did not have the same temporal accuracy as the fully-implicit method. The peak velocities computed here compare favorably with the velocities reported in Table I in [100] for a 40×40 grid despite the fact that a grid with 20×20 elements (21×21 nodes) was used for these computations. Stream-function, temperature, velocity and vorticity contour plots are shown in Figures 10.12 – 10.16.

```
 initialization completed

       0 t 0.0000E+00 div(u)= 0.0000E+00 pmn/pmx=-.115E+02/0.115E+02
         (u_i)mn/mx=0.000E+00/0.000E+00 0.000E+00/0.000E+00
             tmn/mx=0.000E+00/0.100E+01
       0 t 0.0000E+00 dt 1.00E-04 write d3plot file

       1 t 1.0000E-04 div(u)= 1.4427E-11 pmn/pmx=-.126E+02/0.126E+02
         (u_i)mn/mx=-.434E-02/0.434E-02 -.927E-02/0.927E-02
             tmn/mx=0.000E+00/0.100E+01
         LTE=  1.7212E-04 DTSF= 1.5000 old dt=  1.0000E-04 new dt=  1.5000E-04
         Min. diff= -7.6871E-04  at node     311 - Y-Velocity
         Max. diff=  7.6870E-04  at node     149 - Y-Velocity

       2 t 2.5000E-04 div(u)= 3.4505E-11 pmn/pmx=-.147E+02/0.147E+02
         (u_i)mn/mx=-.907E-02/0.907E-02 -.192E-01/0.192E-01
             tmn/mx=0.000E+00/0.100E+01
         LTE=  2.0413E-04 DTSF= 1.5000 old dt=  1.5000E-04 new dt=  2.2500E-04
         Min. diff= -8.4254E-04  at node      15 - Y-Velocity
         Max. diff=  8.4254E-04  at node     428 - Y-Velocity

       3 t 4.7500E-04 div(u)= 7.5514E-11 pmn/pmx=-.168E+02/0.168E+02
         (u_i)mn/mx=-.166E-01/0.166E-01 -.347E-01/0.347E-01
             tmn/mx=0.000E+00/0.100E+01
         LTE=  2.9651E-04 DTSF= 1.4996 old dt=  2.2500E-04 new dt=  3.3742E-04
         Min. diff= -1.3321E-03  at node     407 - Y-Velocity
         Max. diff=  1.3321E-03  at node      38 - Y-Velocity
         .
         .
         .

     150 t 1.9732E+01 div(u)= 8.3505E-10 pmn/pmx=-.103E+03/0.875E+02
         (u_i)mn/mx=-.367E+01/0.367E+01 -.367E+01/0.367E+01
            tmn/mx=0.000E+00/0.100E+01
         LTE=  1.0034E-03 DTSF= 0.9989 old dt= 1.4461E-01 new dt=  1.4445E-01
         Min. diff= -6.7904E-03  at node 139 - Y-Velocity
         Max. diff=  6.7904E-03  at node 322 - Y-Velocity

     151 t 1.9876E+01 div(u)= 8.3491E-10 pmn/pmx=-.103E+03/0.875E+02
         (u_i)mn/mx=-.367E+01/0.367E+01 -.367E+01/0.367E+01
             tmn/mx=0.000E+00/0.100E+01
         LTE=  1.0034E-03 DTSF= 0.9989 old dt= 1.4445E-01 new dt=  1.4429E-01
         Min. diff= -6.7883E-03  at node 322 - Y-Velocity
         Max. diff=  6.7883E-03  at node 139 - Y-Velocity
```

Figure 10.10: Screen output of RMS divergence, minimum/maximum pressure, velocity, temperature and time-step control monitors using the Adams-Bashforth/Trapezoid rule time-integrator.

Algorithm	Δt_{min}	Δt_{max}	u_{max}	v_{max}
De Vahl Davis [100] (40×40 grid)	N/A	N/A	3.634	3.679
Explicit Algorithm	2.00e-4	2.00e-4	3.619	3.643
Semi-Implicit Algorithm	1.00e-4	6.92e-3	3.629	3.635
Fully-Implicit Algorithm	1.00e-4	1.76e-3	3.666	3.674

Table 10.1: Maximum velocities for the 21×21 grid using a) the explicit method, b) semi-implicit projection, and c) the fully-implicit projection method with the AB2/TR time-integrator.

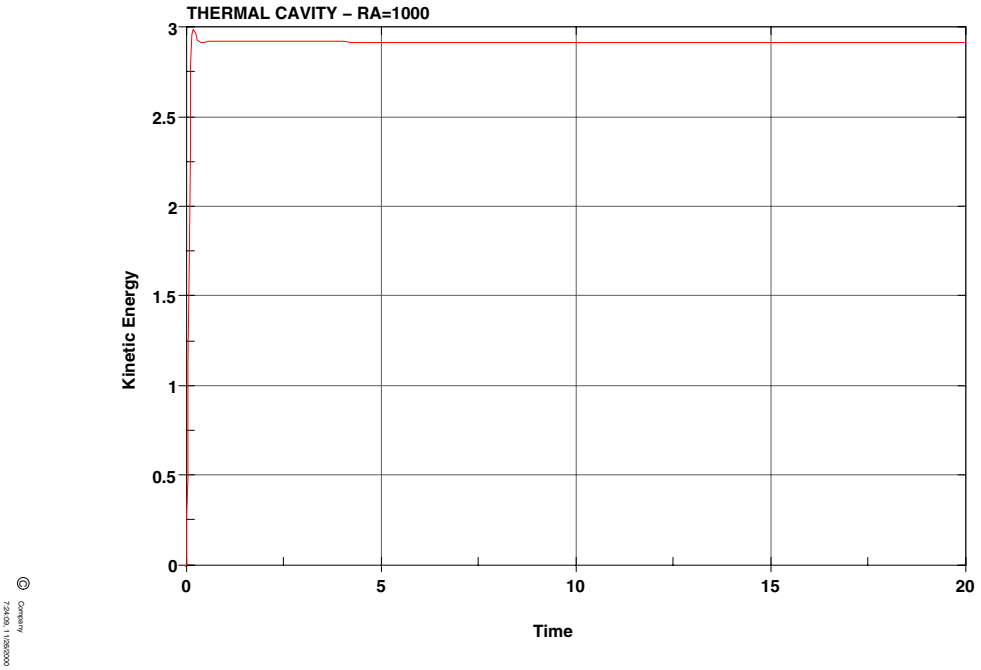


Figure 10.11: Thermal cavity kinetic energy time history.

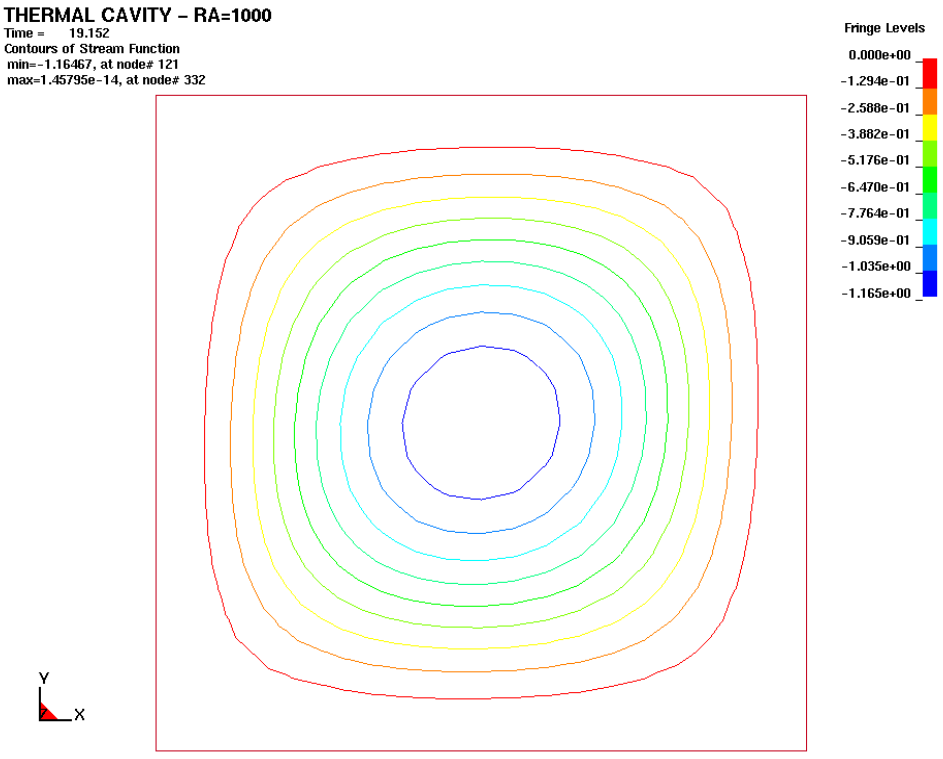


Figure 10.12: Thermal cavity stream function contours.

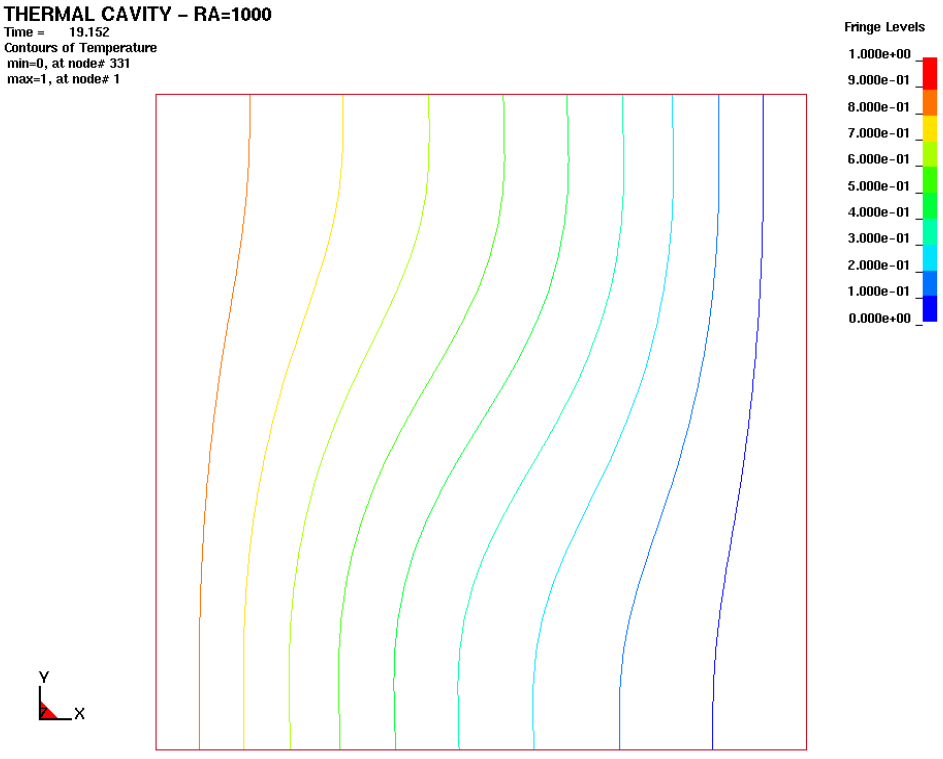


Figure 10.13: Thermal cavity temperature contours.

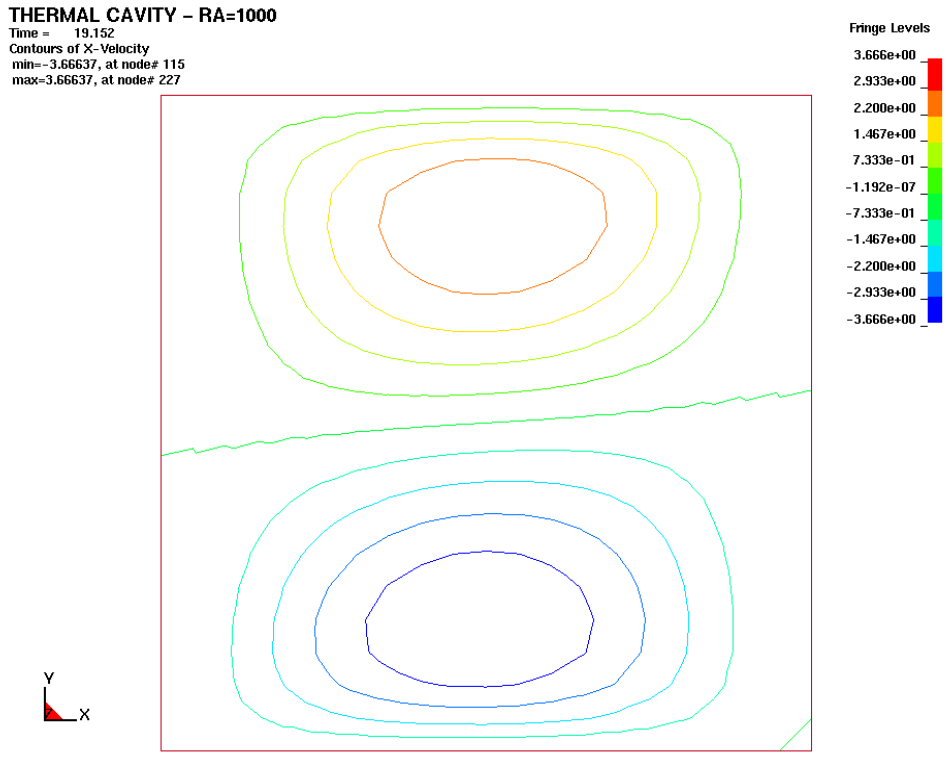


Figure 10.14: Thermal cavity x-velocity contours.

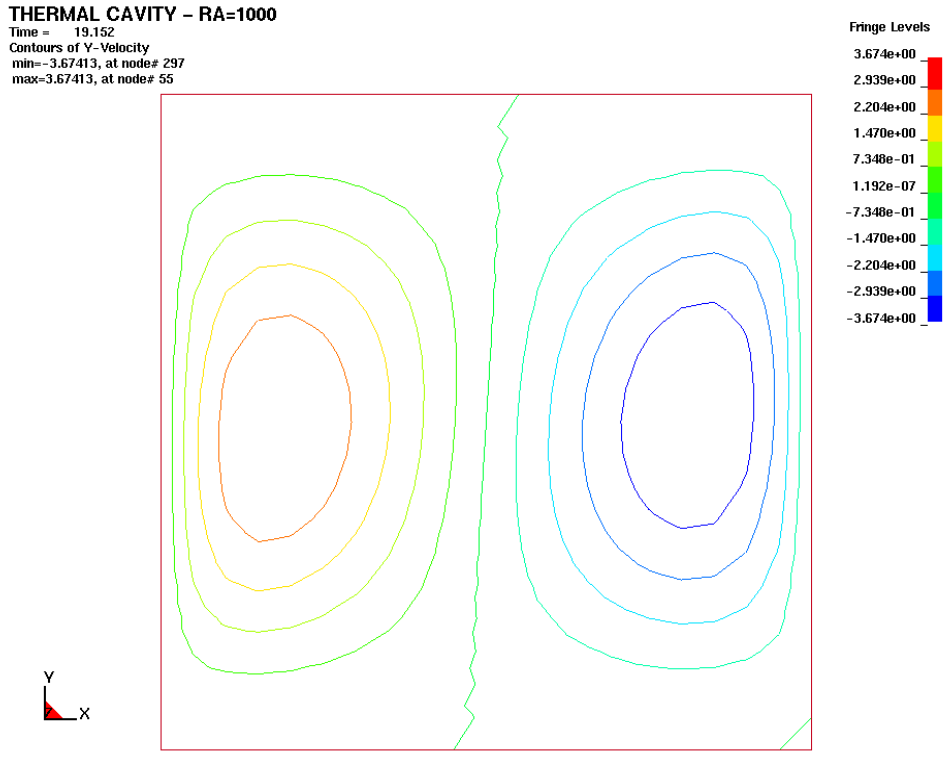


Figure 10.15: Thermal cavity y-velocity contours.

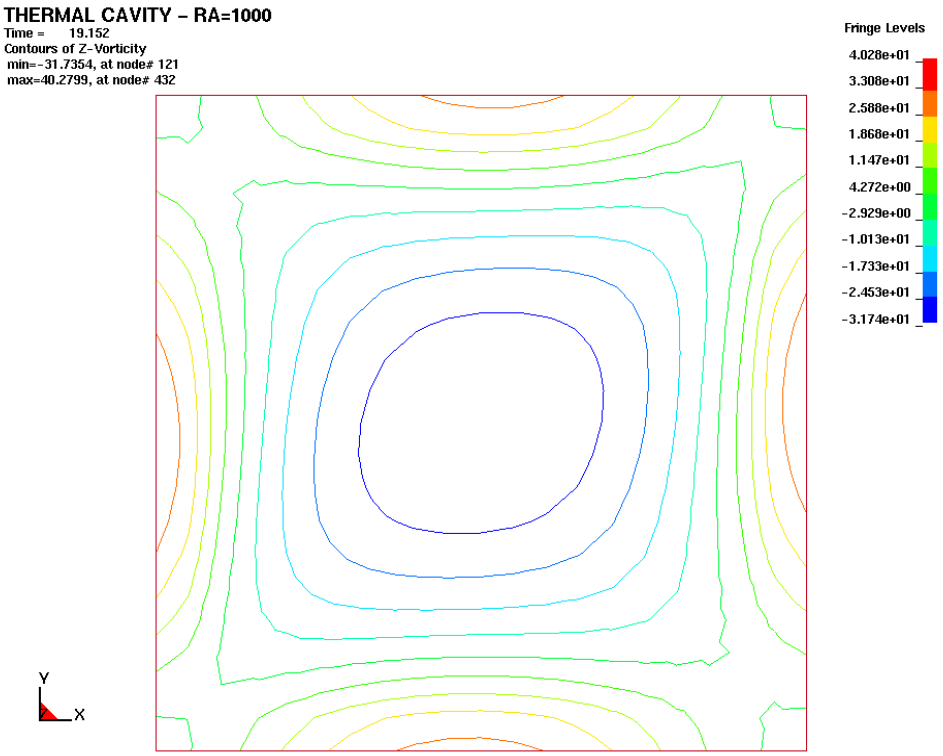


Figure 10.16: Thermal cavity z-vorticity contours.

10.3 The Momentum-Driven Jet

This computation shows the starting vortical structure associated with a heated slot jet entering a relatively cold, quiescent fluid and a classical shear instability phenomena known as the Kelvin-Helmholtz instability. The parameters for the momentum-driven jet are prescribed so that the Reynolds number is $Re = 3561$ based on a 15 mm slot width and $Fr = 326$ ($Fr = v^2/g\beta\Delta T L_c$ where $g = 9.81\ m/s^2$, $\beta\Delta T = 1$, $L_c = 15\ mm$, and $v = 4\ m/s$). Here, the relatively large Froude number indicates that the influence of buoyancy forces is small compared to the inertial forces, i.e., a momentum driven jet.

The grid for the momentum-driven jet is shown in Figure 10.17 where a single plane of symmetry along $x = 0$ is used with the jet half-width $H/2$. In this computation, an energy equation is solved in conjunction with the Navier-Stokes equations using a Boussinesq fluid, i.e., air. The initial conditions consist of an initial div-free velocity field with a free-field temperature of $300K$ and an inlet air jet temperature of $400K$. The working fluid is air with a near-unit Prandtl number $Pr = 0.71$ resulting in a Peclet number $Pe = 2528$ where $Pe = RePr$.

The keyword file `momjet_p2.k` is shown below. In this problem, a fixed CFL=1 was prescribed with the automatic time-step selection based on the CFL number. Due to the effect of the buoyancy terms the vertical jet velocity increased beyond the prescribed 4 m/s so the interval to check and adjust the time is set to `ickdt=2`. A time-accurate simulation is of interest here, so the default second-order time-weighting parameters are used. The sharp gradients in the temperature and velocity fields requires the use of advective flux limiters so `ifct=1` for the momentum and scalar transport equations. The pressure is solved with the SSOR preconditioned conjugate-gradient method (`ipsol=22`) and 5 A-conjugate projection vectors `nvec=5`.

Representative results for the momentum-driven jet are shown in Figures 10.18 – 10.24. The kinetic energy time-history in Figure 10.18 reveals a nearly linear increase in kinetic energy before the starting vortical structure leaves the outflow boundary at the top of the computational domain. The nodal temperature time history plots in Figure 10.19 show the effect of the monotonicity-preserving advective limiter on the temperature where a step-change in the temperature is convected vertically during the starting transient. The nodal time-history plots in Figure 10.20 show a nearly periodic multi-frequency response that is characteristic of the vortex merging process in the shear layer. Snapshots of velocity, temperature and pressure contours are shown in Figures 10.21 – 10.24.

Example: `momjet_p2.k`

```
*KEYWORD
*TITLE
Momentum driven jet
$
$$$
```

```
         2
*CONTROL_CFD_GENERAL
$    insol    dtinit       cfl     ickdt    istats    tstart      iavg
         3    1.0e-4       1.0         2
*CONTROL_CFD_MOMENTUM
$    imass    iadvec      ifct      divu    thetak    thetaa    thetaf
         3        10         1    1.0e-7
$     msol     maxit    ichkit      iwrt     ihist       eps       ihg       ehg
        20       100         2         0         0
*CONTROL_CFD_PRESSURE
$    ipsol     maxit     ickit      iwrt     ihist       eps
        22       300         2         0         0    1.0e-5
$     nvec      stab      beta
         5
*CONTROL_CFD_TRANSPORT
$    itemp      iden     nspec     ifrac
         1         0         0         0
$    imass    iadvec      ifct    thetak    thetaa    thetaf
         3        10         1
$    itsol     maxit    ichkit      iwrt     ihist      eps        ihg       ehg
                                       0
*CONTROL_TERMINATION
$   endtim    endcyc     dtmin     endeng   endmas
   5.00000      1500
$
$
$$$
```

```
        20       102                   2 0.0000000
*BOUNDARY_PRESCRIBED_CFD_SET
$     nsid       dof                lcid        sf
        20       101                   2 1.0000000
$
$ no x-velocity on symmetry plane using nodeset 60
*BOUNDARY_PRESCRIBED_CFD_SET
$     nsid       dof                lcid        sf
        60       101                   2 1.0000000
$
$ jet inlet velocity on nodeset 20
*BOUNDARY_PRESCRIBED_CFD_SET
$     nsid       dof                lcid        sf
        10       102                   1 4.0000000
$ jet inlet temperature
*BOUNDARY_PRESCRIBED_CFD_SET
$     nsid       dof                lcid        sf
        10       104                   3 1.0000000
$ wall temperature
*BOUNDARY_PRESCRIBED_CFD_SET
$     nsid       dof                lcid        sf
        20       104                   1 300.00000
$
$ outflow boundaries - right side, top
*BOUNDARY_OUTFLOW_CFD_SET
$     ssid
        35
*BOUNDARY_OUTFLOW_CFD_SET
$     ssid
        45
*BOUNDARY_OUTFLOW_CFD_SET
$     ssid
        55
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ Setup fluid properties
$
*MAT_CFD_CONSTANT
$      mid        ro        mu         k        Cp      beta      Tref
         1    1.1770 1.9832e-5 2.6240e-2 1.0060e+3   3.33e-3     300.0
$       gx        gy        gz     diff1     diff2     diff3     diff4     diff5
         0     -9.81         0
$    diff6     diff7     diff8     diff9    diff10

$
$ Section & part
$
*SECTION_SHELL
$      sid    elform      shrf       nip     propt   qr/irid     icomp
         1        31
$       t1        t2        t3        t4      nloc
 1.0000000 1.0000000 1.0000000 1.0000000 0.0000000 0.0000000
```

```
*PART
Fluid Domain
$      pid     secid       mid     eosid      hgid      grav    adpopt      tmid
         1         1         1         0         0         0         0         0
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ Handle output of state, time-history,  and restart data
$
*DATABASE_BINARY_D3PLOT
$  dt/cycl      lcdt      beam     npltc
 1.0e-2
*DATABASE_BINARY_D3THDT
$  dt/cycl      lcdt      beam     npltc
 1.0e-6
*DATABASE_HISTORY_NODE
$      nd1       nd2       nd3       nd4       nd5       nd6       nd7       nd8
        11        27        51        83       289
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ Input the mesh, node & segment sets
*INCLUDE
momjet_dat.k
*END
```

Example: `momjet_p2.k`

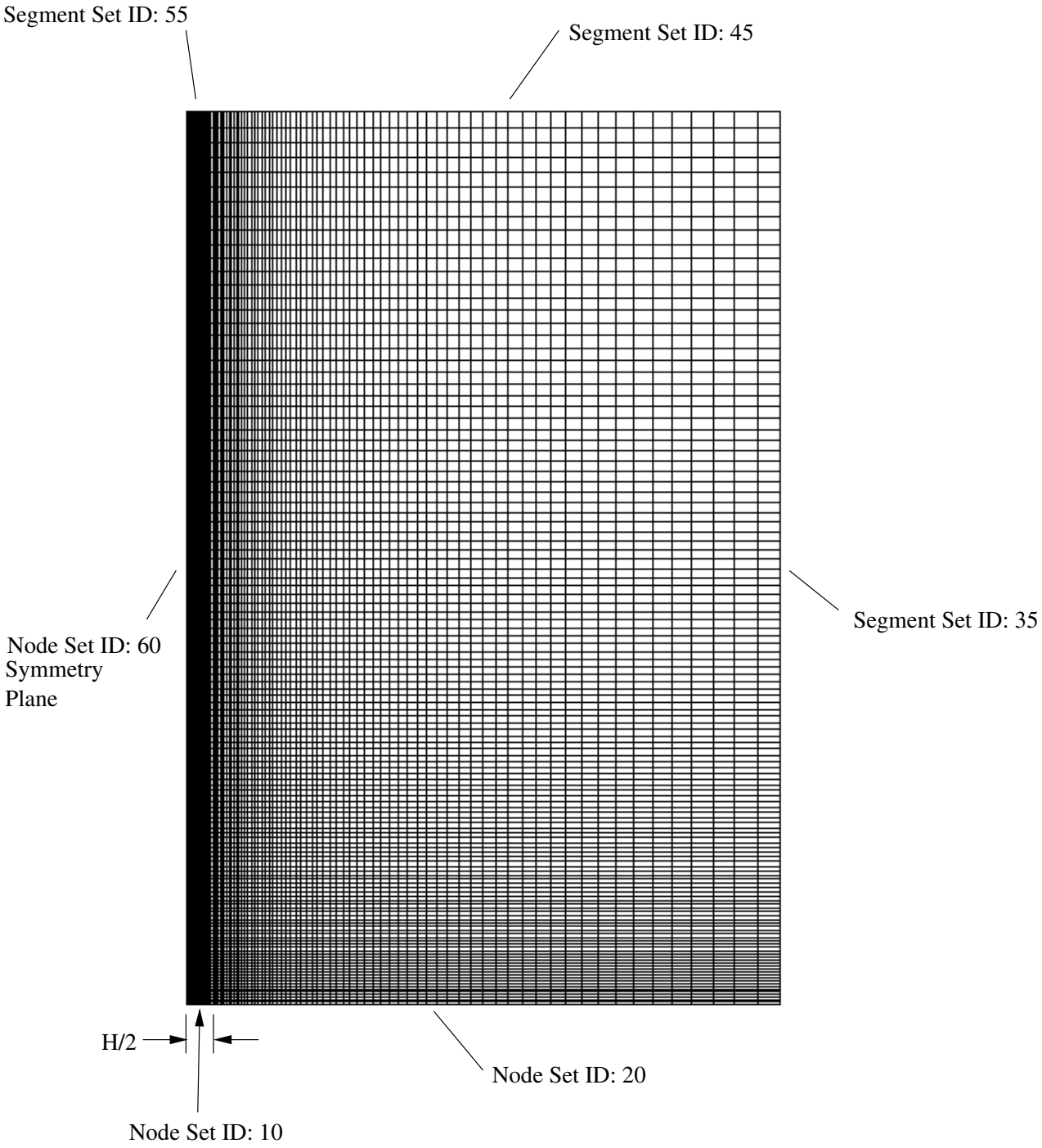


Figure 10.17: Momentum-driven jet mesh with 11250 elements and 11466 nodes.

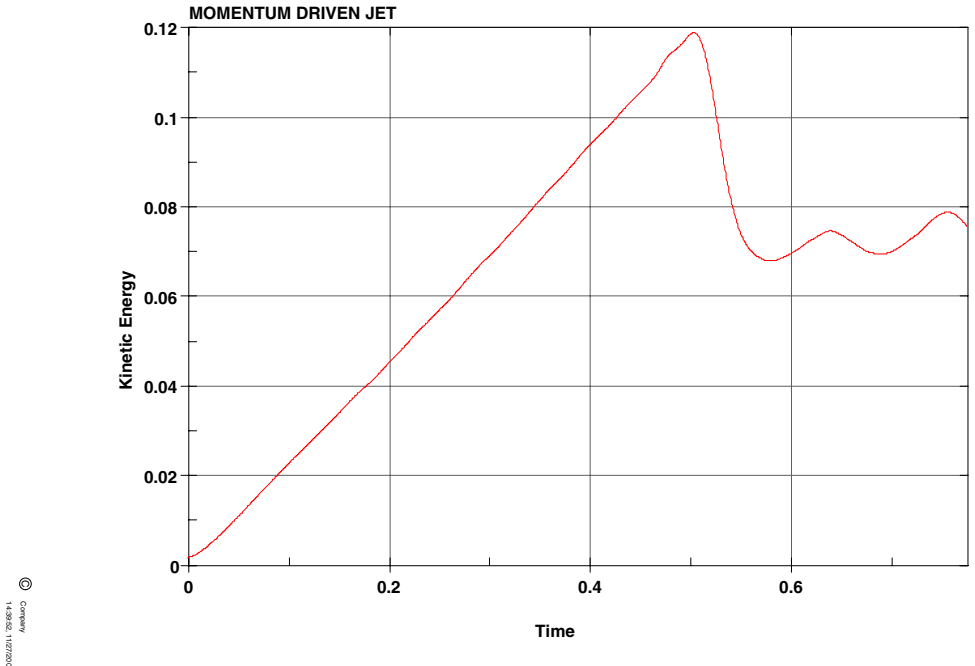


Figure 10.18: Time-history plot of kinetic energy.

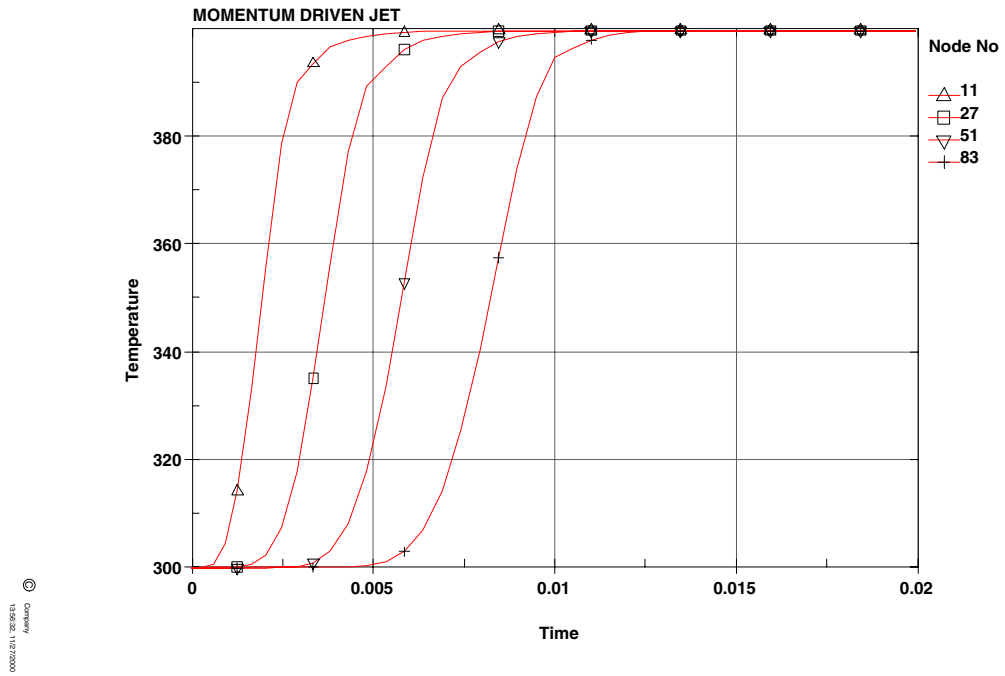
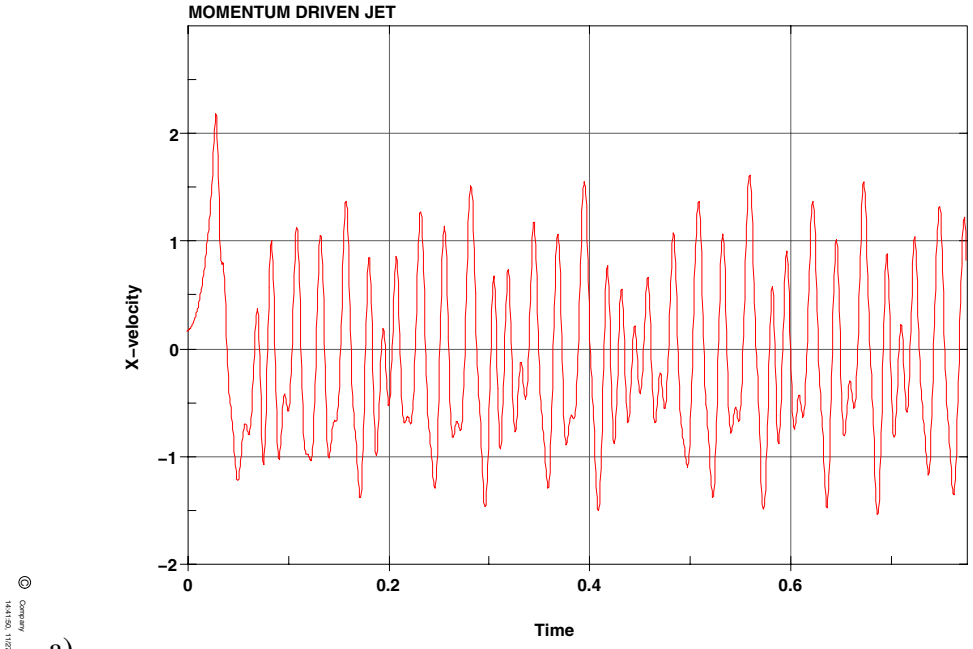
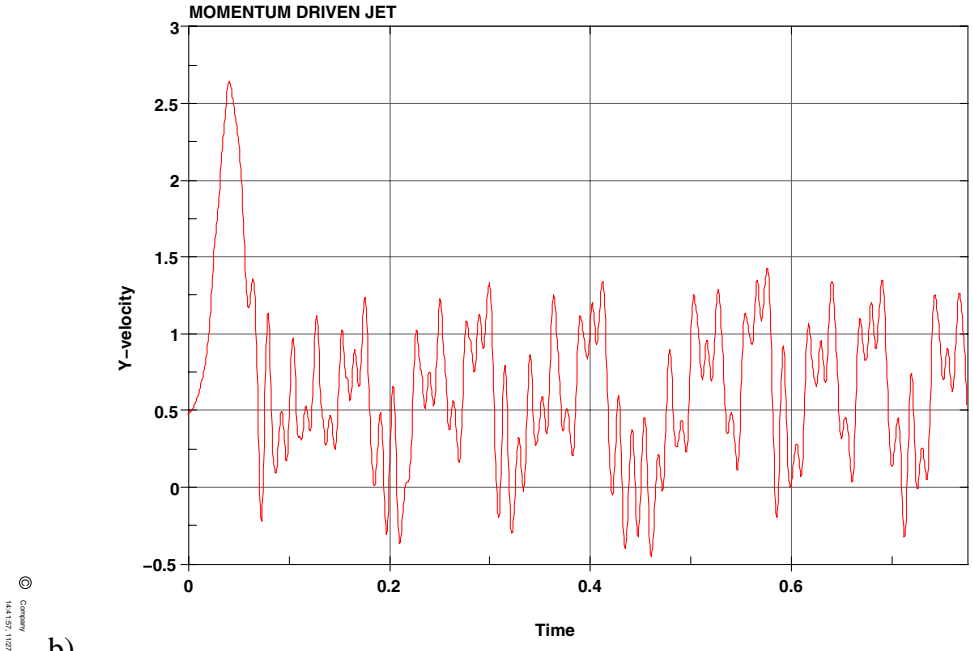


Figure 10.19: Nodal time-history plot of temperature along the jet axis for $0 \leq t \leq 0.02$ s at node-11 (x,y) =(7.50e-4,6.31e-3), node-27 (x,y) =(7.50e-4,1.07e-2), node-51 (x,y) =(7.50e-4,1.52e-2), and node-83 (x,y) =(7.50e-4,1.98e-2).



a)



b)

Figure 10.20: Nodal time-history plot of a) x-velocity and b) y-velocity for node 289 at (x,y) =(1.34e-2,3.71e-2).

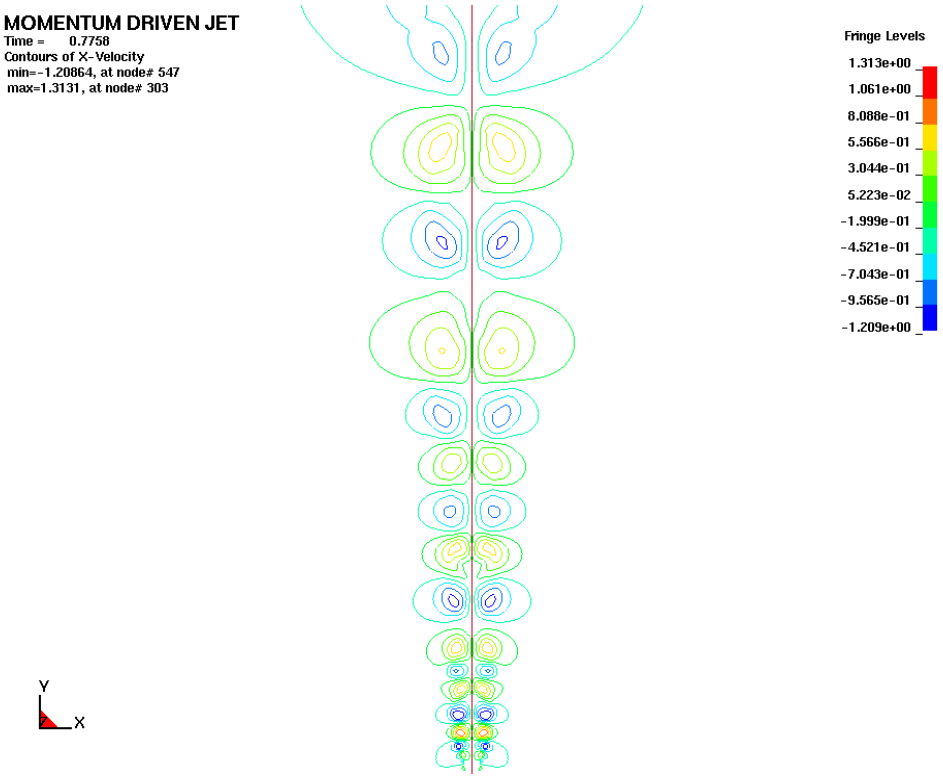


Figure 10.21: X-velocity snapshot at $t = 0.7758$ s.

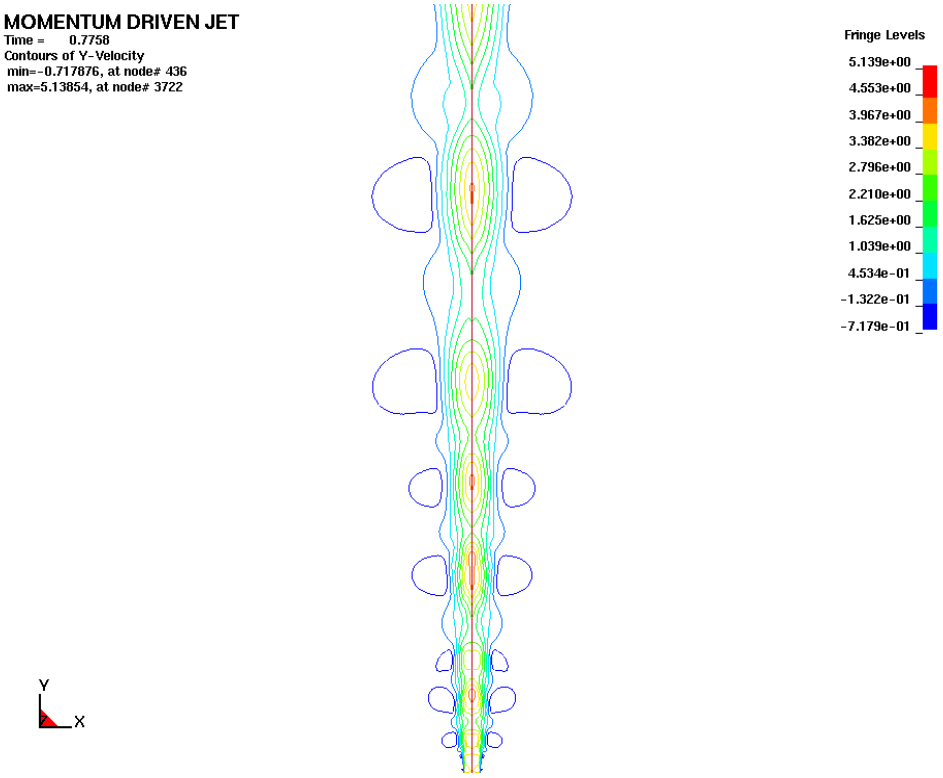


Figure 10.22: Y-velocity snapshot at $t = 0.7758$ s.

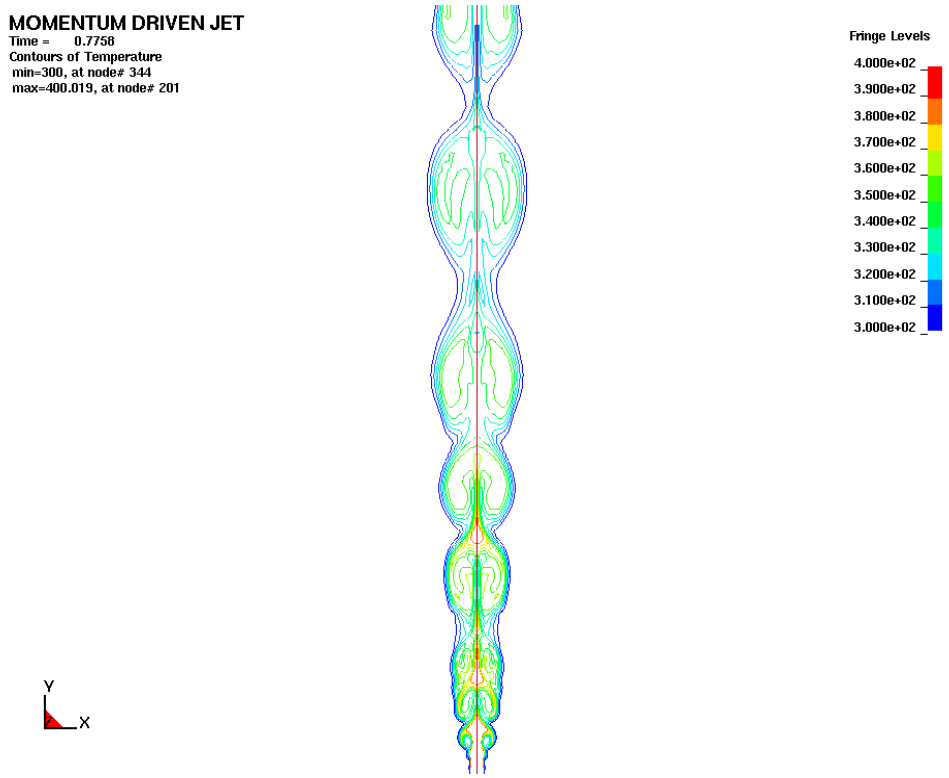


Figure 10.23: Temperature snapshot at $t = 0.7758\ s$.

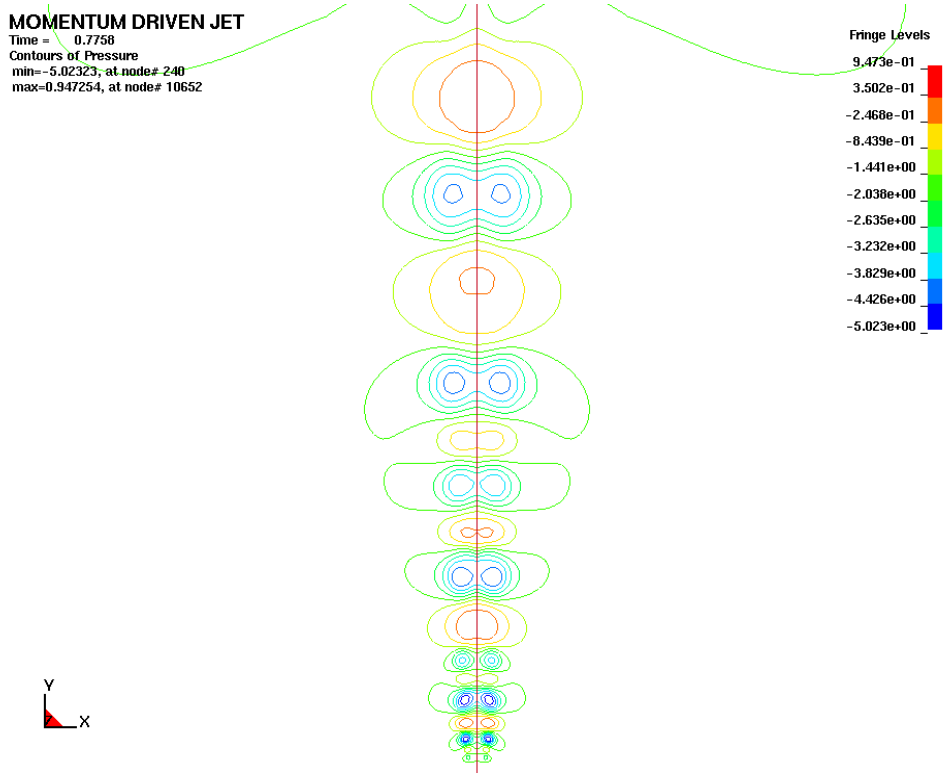


Figure 10.24: Pressure snapshot at $t = 0.7758\ s$.

10.4 The Sheath-Flow Chamber

Recent advancements in micro-machining has permitted the development of portable particle analyzers that sort and categorize particles, e.g., blood cells, at rates exceeding 5000 particles per second. At the core of these devices is a fluid-dynamic focusing chamber commonly referred to as a sheath-flow chamber. The sheath geometry effects the flow patterns which in turn effects the particle trajectory through the sheath and into the capillary tube used to count and categorize the particles. In the ideal situation, the sheath-flow is steady, laminar and void of recirculation zones and swirl effects.

This example presents the analysis of the flow-field in a proposed sheath chamber manufactured from etched silicon wafers. The geometry for the sheath chamber is shown in Figure 10.25. In this configuration, the inlet and outlet axial are square sections with an edge length of 100 μm and a hydraulic diameter of 100 μm. The overall length of the device is 1250 μm with a 100 μm thickness. Axial velocities ranging from 1 m/s to 10 m/s are typical when water is used as the sheath fluid. In this analysis, the properties of water at room temperature are density $\rho = 998\ kg/m^3$ and dynamic viscosity $\mu = 10^-3\ kg/m/s$.

The boundary conditions for the sheath chamber consist of no-slip and no-penetration conditions on the surface of the chamber. The pressure at the axial inlet is prescribed to be 10 kPa, while the side inlet ports have a prescribed pressure of 7.5 kPa. For each inlet port, the prescribed tangential velocity components are zero. At the axial outlet from the chamber, natural (do-nothing) boundary conditions are used. The initial conditions for the problem consist of the initial pressure difference and velocities $u = v = w = 0$.

The `sheath_p2.k` keyword input file is shown below. In this problem, a fixed maximum CFL number is used with the time-step being updated every two time-steps as indicated by `ickdt=2`. Because a steady-state solution is anticipated, 10 A-conjugate base vectors are used in the pressure solution strategy which yields pressure solutions in an average of 4 iterations per time-step.

Although a steady-state, laminar flow solution is expected in this problem, a time-accurate calculation is performed to test the steady-state hypothesis. Figure 10.26 shows the kinetic energy time-history plot for the sheath-chamber and illustrates the trend towards steady conditions for $t > 0.005\ s$. Similar results are observed for the velocity time history plots in Figure 10.27 and 10.28. For $t > 0.005\ s$ the peak x-velocity at the outlet is approximately 4 times larger than along the center line of the axial inlet.

In addition to approaching a steady-state, the flow field is also symmetric about the x-y plane as shown by the pressure isosurfaces in Figure 10.29. In contrast, the y-velocity field is skew-symmetric about the x-y plane as shown by the time-history plot in Figure 10.28. The isosurfaces of helicity $\mathbf{u} \cdot \omega$ shown in Figure 10.30 highlights the rotational structure of the velocity field at the outlet plane. Here, there are four counter-rotating cells – each pair originating at the side inlet port.

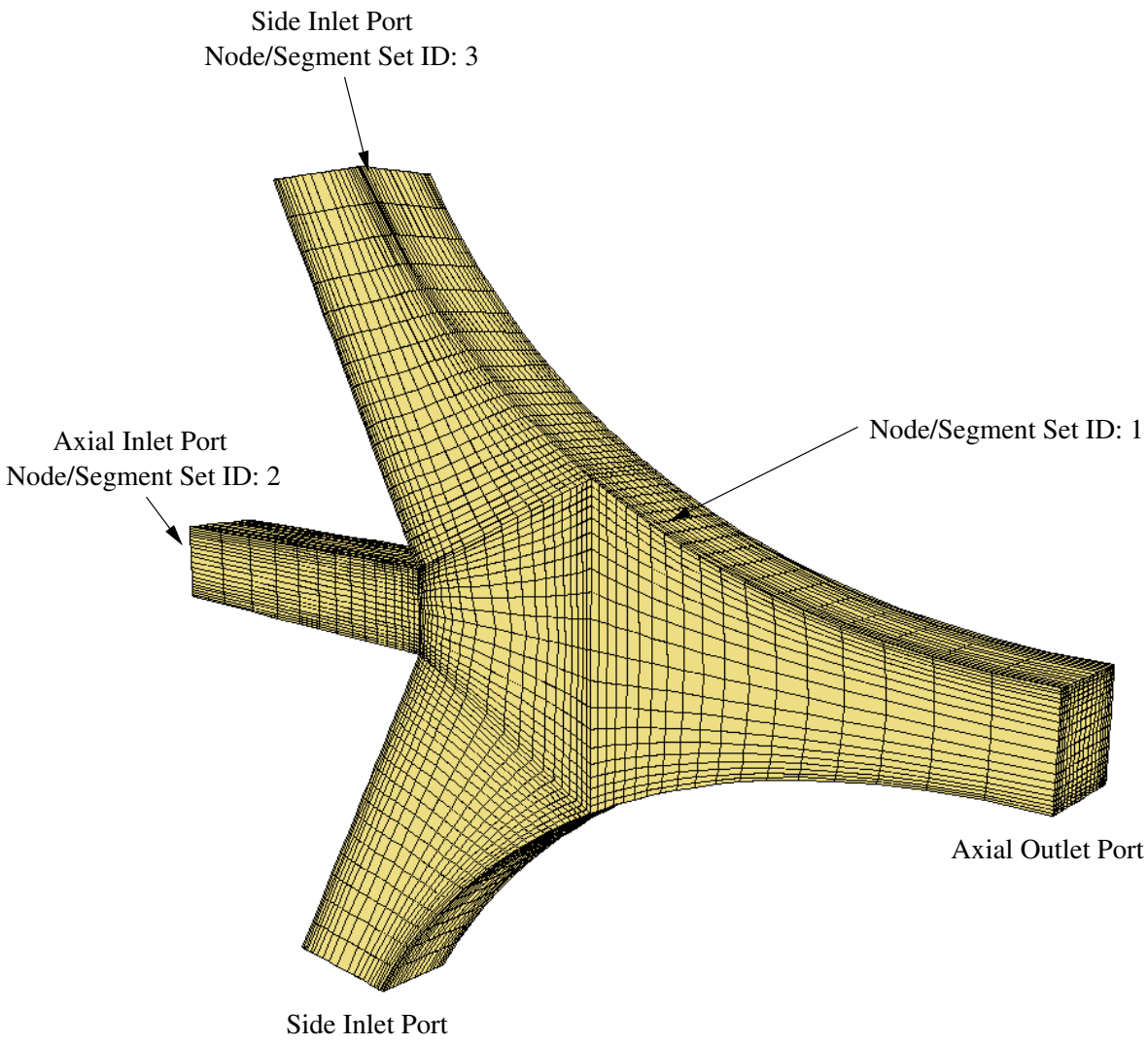


Figure 10.25: Sheath chamber mesh with 33088 elements and 37196 nodes.

Example: `sheath_p2.k`

```
*KEYWORD
*TITLE
Prototype sheath flow chamber
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ Setup the CFD problem - soln=4
$
*CONTROL_SOLUTION
$     soln
         4
*CONTROL_CFD_GENERAL
$    insol    dtinit       cfl     ickdt    istats    tstart      iavg
         3    1.0e-6       4.0         2
*CONTROL_CFD_MOMENTUM
```

```
$    imass     advec      ifct      divu    thetak    thetaa    thetaf
         1        10         1    1.0e-7
$     msol     maxit    ichkit      iwrt     ihist       eps       ihg       ehg
        20       100         2         0         0                 1       1.0
*CONTROL_CFD_PRESSURE
$    ipsol     maxit     ickit      iwrt     ihist       eps
        22       250         2         1         0   1.0e-05
$     nvec      stab      beta
        10
*CONTROL_TERMINATION
$   endtim    endcyc     dtmin     endeng   endmas
1500.00000      2000
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ Define the load curves
$
*DEFINE_CURVE
$     lcid      sidr       sfa       sfo      offa      offo    dattyp
         1
                 0.0                 1.0
              9999.0                 1.0
*DEFINE_CURVE
$     lcid      sidr       sfa       sfo      offa      offo    dattyp
         2
                 0.0                 0.0
              9999.0                 0.0
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ Setup the boundary conditions
$
$ nsid=1 : no-slip, no-penetration surface
$ nsid=2 : on-axis inlet
$ nsid=3 : top/bottom inlet
$ ssid=4 : no-slip, no-penetration surface
$ ssid=5 : on-axis inlet surface
$ ssid=6 : top/bottom surfaces
$
$ Sheath chamber walls -- no-slip, no-penetration
*BOUNDARY_PRESCRIBED_CFD_SET
$     nsid       dof               lcid        sf
         1       101                  2 1.0000000
*BOUNDARY_PRESCRIBED_CFD_SET
$     nsid       dof               lcid        sf
         1       102                  2 1.0000000
*BOUNDARY_PRESCRIBED_CFD_SET
$     nsid       dof               lcid        sf
         1       103                  2 1.0000000
$
$ x-inlet surface
*BOUNDARY_PRESCRIBED_CFD_SET
```

```
$     nsid       dof                lcid        sf
         2       102         0         2 0.0000000
*BOUNDARY_PRESCRIBED_CFD_SET
$     nsid       dof                lcid        sf
         2       103         0         2 0.0000000
$
$ top/bottom x-z velocity
*BOUNDARY_PRESCRIBED_CFD_SET
$     nsid       dof                lcid        sf
         3       101         0         2 0.0000000
*BOUNDARY_PRESCRIBED_CFD_SET
$     nsid       dof                lcid        sf
         3       103         0         2 0.0000000
$
$ Prescribed inlet pressure
*BOUNDARY_PRESSURE_CFD_SET
$     ssid      lcid         p
         2         1   11000.0
*BOUNDARY_PRESSURE_CFD_SET
$     ssid      lcid         p
         3         1   10000.0
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ Setup fluid properties
$
*MAT_CFD_CONSTANT
$      mid        ro        mu         k        Cp      beta      Tref
         1    1000.0    1.0e-3
$       gx        gy        gz     diff1     diff2     diff3     diff4     diff5

$    diff6     diff7     diff8     diff9    diff10

$
$ Section & part
$
*SECTION_SOLID
         1        31
*PART
Fluid domain
         1         1         1         0         0         0         0         0
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ Handle output of state, time-history,  and restart data
$
$
*DATABASE_BINARY_D3PLOT
5.0e-4
*DATABASE_BINARY_D3THDT
1.0e-6
*DATABASE_HISTORY_NODE
```

```
$      nd1       nd2       nd3       nd4       nd5       nd6       nd7       nd8
     17400     17427     17397     18989     35693      6842     25474         9
*DATABASE_BINARY_RUNRSF
      2000
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$ Input the mesh & node sets
$
*INCLUDE
sheath_dat.k
$
*END
```

Example: `sheath_p2.k`

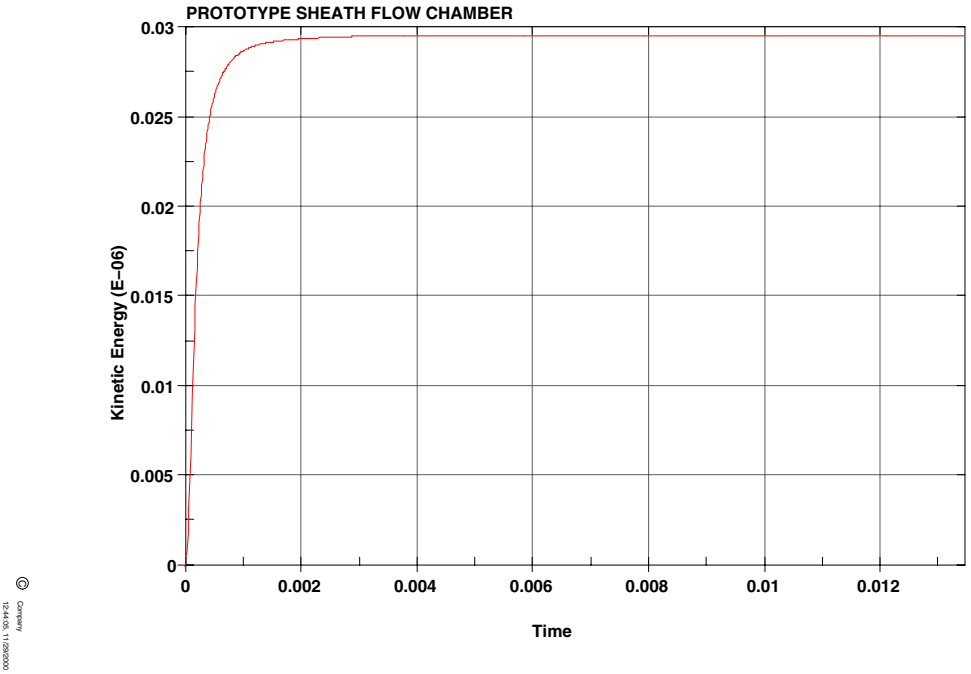


Figure 10.26: Time-history plot of kinetic energy.

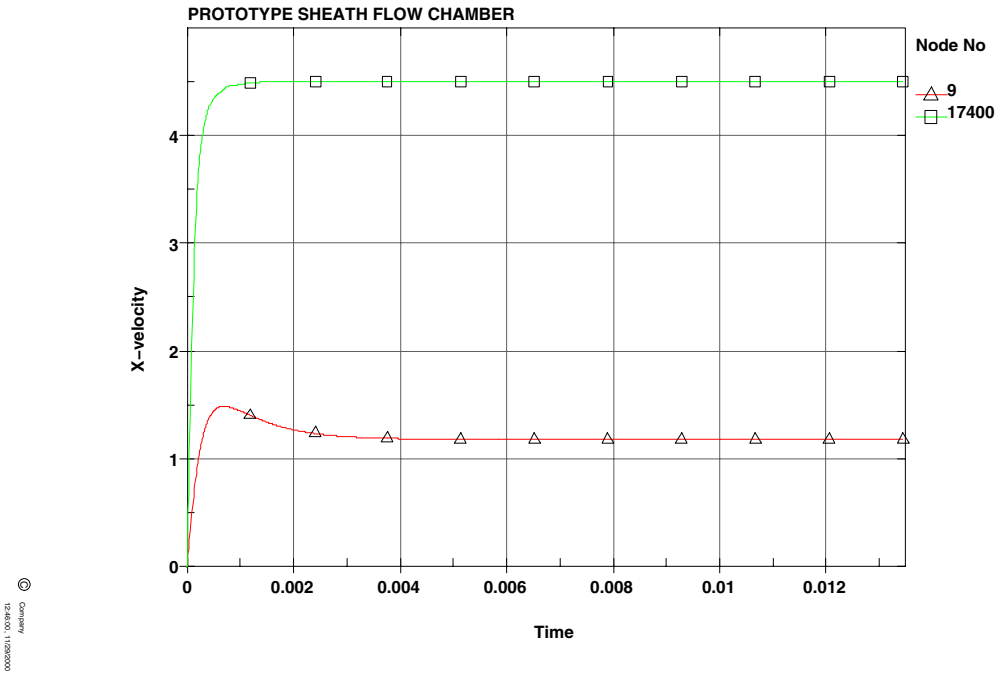


Figure 10.27: Time-history plot of x-velocity at the centerline of the inlet/outlet planes.

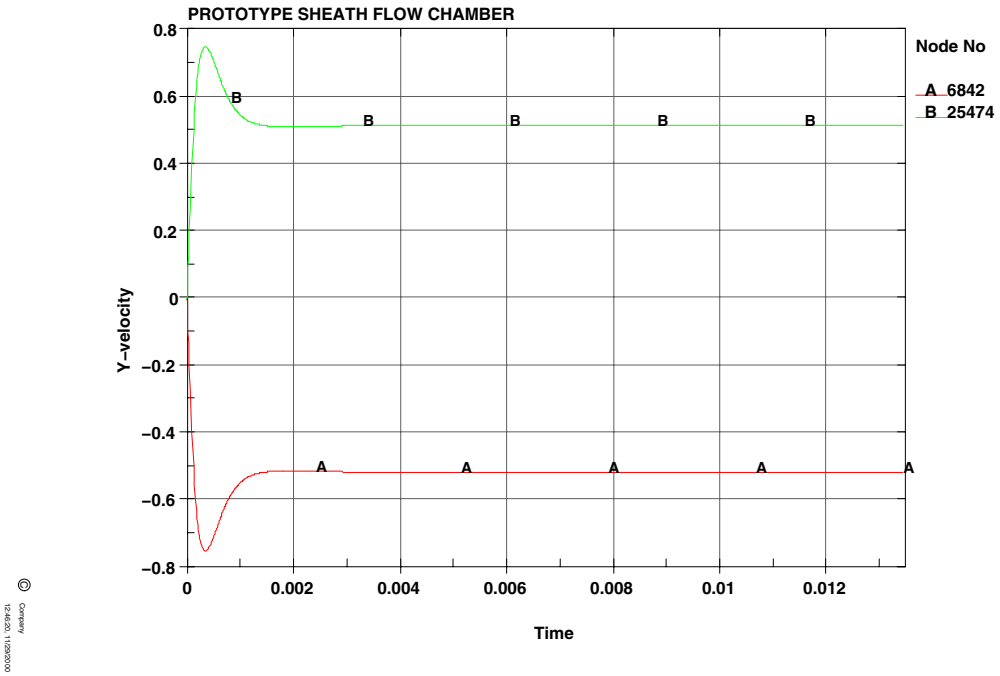


Figure 10.28: Time-history plot of y-velocity at the centerline of the side inlet planes.

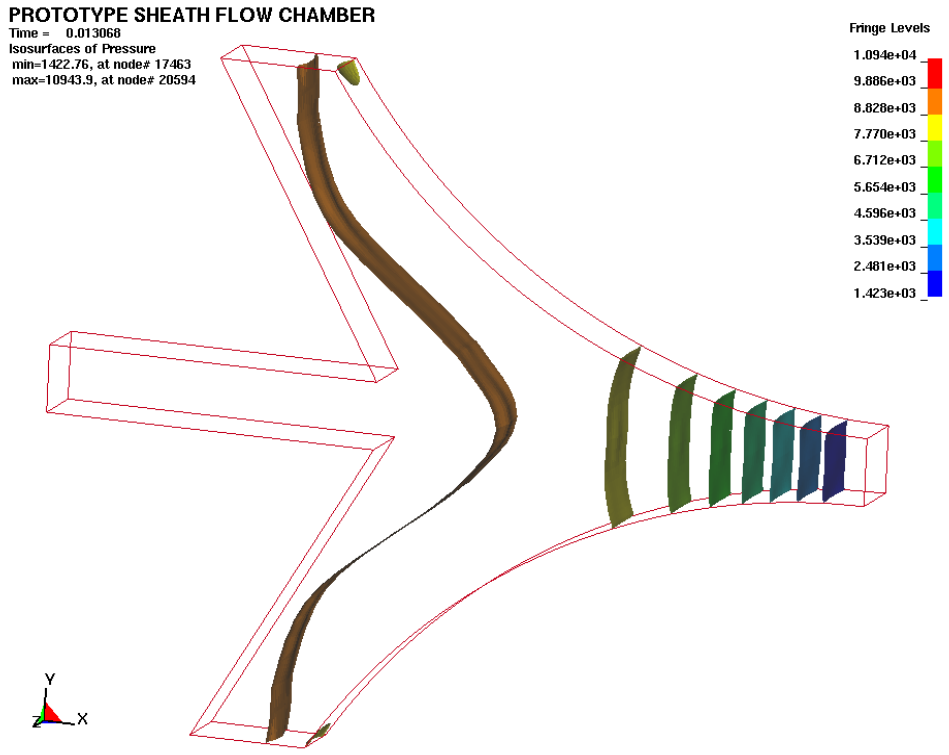


Figure 10.29: Pressure isosurfaces at $t = 0.013\ s$.

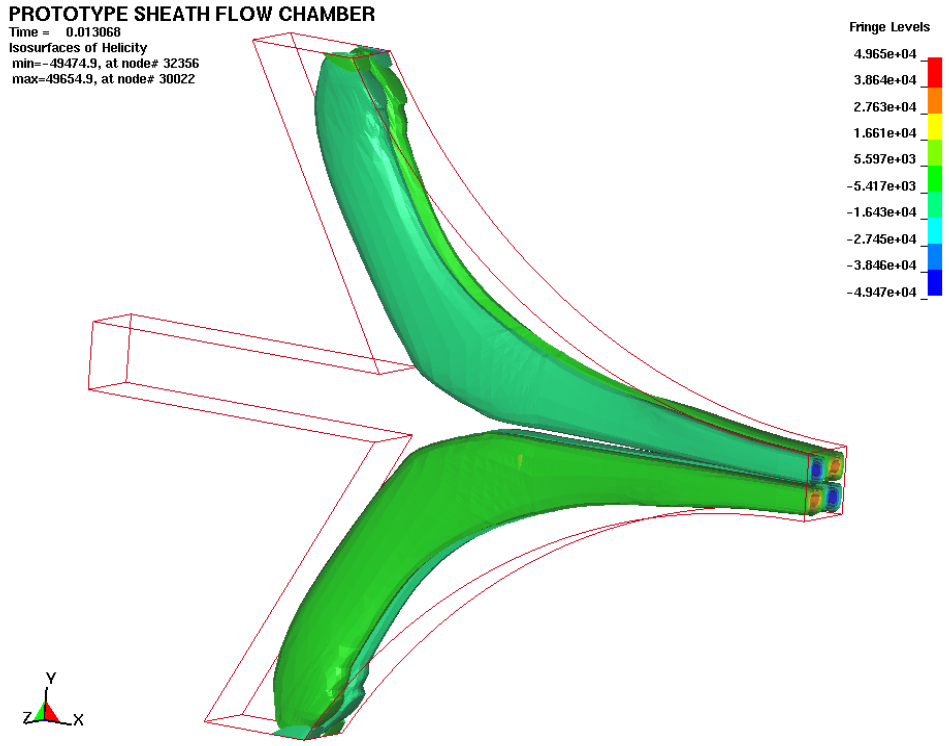


Figure 10.30: Helicity isosurfaces at $t = 0.013\ s$.

Appendix A

Sense Switch Controls

The status of an LS-DYNA calculation can be determined by typing "^C" (control-c). This sends an interrupt to LS-DYNA which is trapped and the user is prompted to input a sense witch code. For the incompressible flow solver, the sw2., sw5., and swa. switches are ignored. The active flow-solver sense switches are:

Type	Response
sw0.	LS-DYNA terminates immediately.
sw1.	A restart file is written and LS-DYNA terminates.
sw3.	A restart file is written and LS-DYNA proceeds.
sw4.	A plot state is written and LS-DYNA proceeds.
div	Toggle the output of the divergence metrics for a projection step.
grid	Toggle the output of the grid Reynolds and CFL numbers.
iter	Toggle the output of statistics for the iterative equation solvers.
lprint	Toggle the output of velocity min/max.
quit	LS-DYNA terminates immediately.
stop	LS-DYNA terminates immediately.

Bibliography

[1] Philip M. Gresho, Stevens T. Chan, Robert L. Lee, and Craig D. Upson. A modified finite element method for solving the time-dependent, incompressible navier-stokes equations. part 1: Theory. *International Journal for Numerical Methods in Fluids*, 4:557–598, 1984.

[2] Philip M. Gresho, Stevens T. Chan, Robert L. Lee, and Craig D. Upson. A modified finite element method for solving the time-dependent, incompressible navier-stokes equations. part 2: Applications. *International Journal for Numerical Methods in Fluids*, 4:619–640, 1984.

[3] Philip M. Gresho. On the theory of semi-implicit projection methods for viscous incompressible flow and its implementation via a finite element method that also introduces a nearly consistent mass matrix. part 1: Theory. *International Journal for Numerical Methods in Fluids*, 11:587–620, 1990.

[4] Philip M. Gresho and Stevens T. Chan. On the theory of semi-implicit projection methods for viscous incompressible flow and its implementation via a finite element method that also introduces a nearly consistent mass matrix. part 2: Implementation. *International Journal for Numerical Methods in Fluids*, 11:621–659, 1990.

[5] P. M. Gresho and R. L. Sani. *Incompressible flow and the finite element method, Advection-diffusion and isothermal laminar flow*. John Wiley & Sons, Chicester, England, 1998.

[6] Philip M. Gresho and Robert L. Sani. On pressure boundary conditions for the incompressible navier-stokes equations. *International Journal for Numerical Methods in Fluids*, 7:1111–1145, 1987.

[7] Philip M. Gresho, Robert L. Lee, and Robert Sani. *On the time-dependent solution of the Navier-Stokes equations in two and three dimensions*, volume 1, page 27. Pineridge Press, Swansea, U.K, 1980.

[8] Thomas J. R. Hughes. *The Finite Element Method*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1987.

[9] O. C. Zienkiewicz and Robert L. Taylor. *The Finite Element Method*, volume 2. McGraw-Hill Book Company Limited, Berkshire, England, fourth edition, 1991.

[10] Philip M. Gresho and Robert L. Sani. *Computational Fluid Dynamics Review '97*, chapter Problems and solutions (generalized and FEM) related to rapid and impulsive changes for incompressible flow. John Wiley and Sons, New York, New York, 1997.

[11] Mark A. Christon and Thomas E. Spelce. Visualization of high resolution three-dimensional nonlinear finite element analyses. In *Visualization '92*, pages 324–331, Boston, MA, 1992. IEEE. (LLNL UCRL-JC-110110).

[12] Mark A. Christon. Visualization methods for high-resolution, transient, 3-D, finite element simulations. In *International Workshop on Visualization*, Paderborn, Germany, January 1994. (*invited paper*, LLNL UCRL-JC-119879).

[13] Gerald L. Goudreau and John O. Halquist. Recent developments in large-scale finite element lagrangian hydrocode technology. *Computer Methods in Applied Mechanics and Engineering*, 33:725, 1982.

[14] T. Belytschko, J. S. Ong, W. K. Liu, and J. M. Kennedy. Hourglass control in linear and nonlinear problems. *Computer Methods in Applied Mechanics and Engineering*, 43:251–276, 1984.

[15] W. K. Liu and T. Belytschko. Efficient linear and nonlinear heat conduction with a quadrilateral element. *International Journal for Numerical Methods in Engineering*, 20:931–948, 1984.

[16] W. K. Liu, J. S. Ong, and R. A. Uras. Finite element stabilization matrices. *Computer Methods in Applied Mechanics and Engineering*, 53:13–46, 1985.

[17] Robert G. Whirley and Bruce E. Engelmann. Dyna3d: A nonlinear, explicit, three-dimensional finite element code for solid and structural mechanics - user manual. Technical Report UCRL-MA-107254, Rev. 1, Lawrence Livermore National Laboratory, 1993.

[18] D. R. Kincaid, T. C. Oppe, and D. M. Young. Itpackv 2d user's guide. Technical report, Center for Numerical Analysis, The University of Texas at Austin, May 1989.

[19] C. Farhat and Michel Lesoinne. Automatic partitioning of unstructured meshes for the parallel solution of problems in computational mechanics. *International Journal for Numerical Methods in Engineering*, 36:745–764, 1993.

[20] A. Pothen, H. Simon, and K. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Matrix Anal. Appl.*, 11:430–452, 1990.

[21] H. D. Simon. Partitioning of unstructured problems for parallel processing. *Computing Systems in Engineering*, 2:135, 1991.

[22] S. T. Barnard and H. D. Simon. A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. In *6th SIAM Conference on Parallel Processing for Scientific Computing*, pages 711–718, 1993.

[23] B. Hendrickson and R. Leland. An improved spectral graph partitioning algorithm for mapping parallel computations. Technical Report SAND92-1460, Sandia National Laboratories Report, September 1992.

[24] B. Hendrickson and R. Leland. Multidimensional spectral load balancing. Technical Report SAND93-0074, Sandia National Laboratories Report, January 1993.

[25] B. Hendrickson and R. Leland. A multilevel algorithm for partitioning graphs. Technical Report SAND93-1301, Sandia National Laboratories Report, October 1993.

[26] B. Hendrickson and R. Leland. The chaco user's guide - version 1.0. Technical Report SAND93-2339, Sandia National Laboratories Report, October 1993.

[27] Livermore Software Technology Corporation. *LS-DYNA Keyword User's Manual – Nonlinear Dynamic Analysis of Structures*. Livermore Software Technology Corporation, 7374 Las Positas Road, Livermore, California 94550, USA, version 960 edition, 2001.

[28] Alexandre Joel Chorin. Numerical solution of the navier-stokes equations. *Mathematics of Computations*, 22:745–762, 1968.

[29] J. Van Kan. A second-order accurate pressure-correction scheme for viscous incompressible flow. *SIAM Journal for Scientific and Statistical Computing*, 7:870–891, 1986.

[30] John B. Bell, Philip Colella, and Harland M. Glaz. A second-order projection method for the incompressible navier-stokes equations. *Journal of Computational Physics*, 85:257–283, 1989.

[31] Philip M. Gresho, Stevens T. Chan, Mark A. Christon, and Allen C. Hindmarsh. A little more on stabilized $q1q1$ for transient viscous incompressible flow. *International Journal for Numerical Methods in Fluids*, 21:837–856, 1995.

[32] Philip M. Gresho and Stevens T. Chan. Projection 2 goes turbulent – and fully implicit. *preprint International Journal for Computational Fluid Dynamics*, March 1996. (LLNL UCRL-JC-123727).

[33] Ann S. Almgren, John B. Bell, Phillip Colella, and Louis H Howell. An adaptive projection method for the incompressible euler equations. In *Eleventh AIAA Computational Fluid Dynamics Conference*, pages 530–539. AIAA, 1993.

[34] Ann S. Almgren, John B. Bell, and William G. Szymcyzk. A numerical method for the incompressible navier-stokes equations based on an approximate projection. *SIAM Journal for Scientific Computing*, 17(2):358–369, March 1996.

[35] Ann S. Almgren, John B. Bell, Phillip Colella, Louis H. Howell, and Michael L. Welcome. A conservative adaptive projection method for the variable density incompressible navier-stokes equations. *Journal of Computational Physics*, 142:1–46, 1998.

[36] William J. Rider. The robust formulation of approximate projection methods for incompressible flows. Technical Report LA-UR-3015, Los Alamos National Laboratory, 1994.

[37] William J. Rider. Filtering nonsolenoidal modes in numerical solutions of incompressible flows. Technical Report LA-UR-3014, Los Alamos National Laboratory, Los Alamos, New Mexico, September 1994.

[38] W. J. Rider, D. B. Kothe, S. J. Mosso, J. H. Cerutti, and J. I. Hochstein. Accurate solution algorithms for incompressible multiphase flows. Technical Report AIAA-95-0699, AIAA, Reno, Nevada, January 1995.

[39] William J. Rider. Approximate projection methods for incompressible flow: implementation, variants and robustness. Technical Report LA-UR-2000, Los Alamos National Laboratory, Los Alamos, New Mexico, July 1995.

[40] Michael L. Minion. A projection method for locally refined grids. *Journal of Computational Physics*, 127:158–178, 1996.

[41] J.-L. Guermond and L. Quartapelle. Calculation of incompressible viscous flow by an unconditionally stable projection fem. *Journal of Computational Physics*, 132:12–23, 1997.

[42] Elbridge G. Puckett, Ann S. Almgren, John B. Bell, Daniel L. Marcus, and William J. Rider. A high-order projection method for tracking fluid interfaces in variable density incompressible flows. *Journal of Computational Physics*, 130:269–282, 1997.

[43] Mark Sussman, Ann S. Almgren, John B. Bell, Phillip Colella, Louis H. Howell, and Michael L. Welcome. An adaptive level set approach for two-phase flows. *Journal of Computational Physics*, 148:81–124, 1999.

[44] Omar M. Knio, Habib N. Najm, and Peter S. Wyckoff. A semi-implicit numerical scheme for reacting flow. ii. stiff operator-split formulation. *pre-print submitted to Journal of Computational Physics*, 1999.

[45] David L. Brown and Michael L. Minion. Performance of under-resolved two-dimensional incompressible flow simulations. *Journal of Computational Physics*, 122:165–183, 1995.

[46] Michael L. Minion and David L. Brown. Performance of under-resoloved two-dimensional incompressible flow simulations, ii. *Journal of Computational Physics*, 138:734–765, 1997.

[47] Brian B. Wetton. Error analysis of pressure increment schemes. *submitted to SINUM*, April 1998.

[48] Jean-Luc Guermond. Some implementations of projection methods for navier-stokes equations. *Mathematical Modelling and Numerical Analysis*, 30(5):637–667, 1996.

[49] Jean-Luc Guermond. A convergence result for the approximation of the navier-stokes equations by an incremental projection method. *C. R. Acad. Sci. Paris*, 325:1329–1332, 1997.

[50] Jean-Luc Guermond and L. Quartapelle. On the approximation of the unsteady navier-stokes equations by finite element projection methods. *Numerische Mathematik*, 80:207–238, 1998.

[51] Jean-Luc Guermond and L. Quartapelle. On stability and convergence of projection methods based on pressure poisson equation. *International Journal for Numerical Methods in Fluids*, 26:1039–1053, 1998.

[52] Ann S. Almgren, John B. Bell, and William Y. Crutchfield. Approximate projection methods: Part i. inviscid analysis. *SIAM Journal on Scientific Computing*, 22(4):1139–1159, 2000.

[53] J. C. Heinrich, S. R. Idelsohn, E. Onate, and C. A. Vionnet. Boundary conditions for finite element simulations of convective flows with artificial boundaries. *International Journal for Numerical Methods in Fluids*, 39:1053–1071, 1996.

[54] Robert L. Sani, P. M. Gresho, Robert L. Lee, and D. F. Griffiths. The cause and cure (?) of the spurious pressures generated by certain fem solutions of the incompressible navier-stokes equations: Part 1. *International Journal for Numerical Methods in Fluids*, 1:17–43, 1981.

[55] Robert L. Sani, P. M. Gresho, Robert L. Lee, D. F. Griffiths, and Michael Engelman. The cause and cure (!) of the spurious pressures generated by certain fem solutions of the incompressible navier-stokes equations: Part 2. *International Journal for Numerical Methods in Fluids*, 1:171–204, 1981.

[56] David Griffiths and David Silvester. Unstable modes of the q1-p0 element. Technical Report NA-257, University of Manchester/UMIST, Manchester Centre for Computational Mathematics, September 1994.

[57] T. J. R. Hughes and L. P. Franca. A mew finite element formulation for computational fluid dynamics: Vii. the stokes problem with various well-posed boundary conditions: symmetric formulations that converge for all velocity/pressure spaces. *Computer Methods in Applied Mechanics and Engineering*, 65:85–96, 1987.

[58] David J. Silvester and N. Kechkar. Stabilised bilinear-constant velocity-pressure finite elements for the conjugate gradient solution of the stokes problem. *Computer Methods in Applied Mechanics and Engineering*, 79:71–86, 1990.

[59] N. Kechkar and D. Silvester. Analysis of locally stabilised mixed finite element methods for the stokes problem. *Mathematics of Computation*, 58:1–10, 1992.

[60] David Silvester. Stabilised mixed finite element methods. *draft preprint*, January 1995.

[61] Sean Norburn and David Silvester. Stabilised vs. stable mixed methods for incompressible flow. Technical Report NA-307, University of Manchester/UMIST, Manchester Centre for Computational Mathematics, September 1997.

[62] Stevens T. Chan and Gayle Sugiyama. User's manual for mc_wind: A new mass-consistent wind model for arac-3. Technical Report UCRL-MA-129067, Lawrence Livermore National Laboratory, Livermore, California 94550, November 1997.

[63] Mark A. Christon. A domain-decomposition message-passing approach to transient viscous incompressible flow using explicit time integration. *Computer Methods in Applied Mechanics and Engineering*, 148:329–352, 1997. (Sandia SAND96-2566J).

[64] Paul F. Fischer. Projection techniques for iterative solution of $ax = b$ with successive right-hand sides. Technical Report 93-90, ICASE, 1993. (Also NASA CR-191571).

[65] Youcef Saad. On the lanczos method for solving symmetric linear systems with several right-hand sides. *Mathematics of Computation*, 48(178):651–662, April 1987.

[66] Tony F. Chan and W. L. Wan. Analysis of projection methods for solving linear systems with multiple right-hand sides. *SIAM Journal on Scientific Computing*, 18(6):1698–1721, November 1997.

[67] T. J. R. Hughes. personal communication. October 1998.

[68] S. C. Eisenstat. Efficient implementation of a class of preconditioned conjugate gradient methods. *SIAM Journal for Scientific and Statistical Computing*, 2(1):1–4, 1981.

[69] J. M. Ortega. Efficient implementations of certain iterative methods. *SIAM Journal for Scientific and Statistical Computing*, 9(5):882–891, September 1988.

[70] S. C. Eisenstat, J. M. Ortega, and C. T. Vaughan. Efficient implementation of a class of preconditioned conjugate gradient methods. *SIAM Journal for Scientific and Statistical Computing*, 11(5):859–872, 1990.

[71] O.O. Storaasli, D.T. Nguyen, and T.K. Agarwal. A parallel-vector algorithm for rapid structural analysis on high-performance computers. Technical Report 102614, NASA, April 1990.

[72] T. K. Agarwal, O. O. Storaasli, and D. T. Nguyen. A parallel-vector algorithm for rapid structural analysis on high-performance computers. *AIAA*, 90-1149, 1990.

[73] P. R. Spalart and S. R. Allmaras. A one-equation turbulence model for aerodynamic flows. In *AIAA-92-0439*, Reno, Nevada, January 1992. AIAA 30th Aerospace Science Meeting and Exhibit.

[74] Barrett S. Baldwin and Timothy J. Barth. A one-equation turbulence transport model for high reynolds number wall-bounded flows. In *AIAA-91-0610*, Reno, Nevada, January 1991. AIAA 29th Aerospace Science Meeting and Exhibit.

[75] W. C. Reynolds. The potential and limitations of direct and large eddy simulations. In J. L. Lumley, editor, *Whither Turbulence? Turbulence at the Crossroads*, pages 313–343, Berlin, March 1989. Springer-Verlag.

[76] David C. Wilcox. *Turbulence Modeling for CFD*. DCW Industries, Inc., La Canada, California, 1993.

[77] Chin-Hoh Moeng. A large-eddy-simulation model for the study of planetary boundary layer turbulence. *Journal of the Atmospheric Sciences*, 41(13):2052–2062, July 1984.

[78] Peter R. Voke and Socrates G. Potamitis. Numerical simulation of a low-reynolds-number turbulent wake behind a flat plate. *International Journal for Numerical Methods in Fluids*, 19:377–393, 1995.

[79] D. Veynante and T. Poinsot. Large eddy simulation of combustion instabilities in turbulent premixed burners. Technical report, Center for Turbulence Research, Stanford University, Palo Alto, California, 1997.

[80] W.-W. Kim and S. Menon. Large eddy simulations of reacting flow in a dump combustor. Technical Report AIAA 98-2432, 29th AIAA Fluid Dynamics Conference, Albuquerque, New Mexico, June 1998.

[81] C. Weber, F. Ducros, and A. Corjon. Large eddy simulation of complex turbulent flows. Technical Report AIAA 98-2651, 29th AIAA Fluid Dynamics Conference, Albuquerque, New Mexico, June 1998.

[82] F. F. Grinstein and C. Fureby. Monotonically integrated large eddy simulation of free shear flows. In *36th Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, January 1998. American Institute of Aeronautics and Astronautics.

[83] D. Kwak and W. C. Reynoldsand J. H. Ferziger. Three-dimensional time dependent computation of turbulent flow. Technical Report TF-5, Thermosciences Division, Department of Mechanical Engineering, Stanford University, Stanford, California, May 1975.

[84] M. Germano. Turbulence: the filtering approach. *Journal of Fluid Mechanics*, 238:325–336, 1992.

[85] J. Smagorinsky. General circulation experiments with the primitive equations, i. the basic experiment. *Monthly Weather Review*, 91(3):99–164, 1963.

[86] Robert S. Rogallo and Parviz Moin. *Annual Review of Fluid Mechanics*, volume 16, chapter Numerical simulation of turbulent flows, pages 99–137. Annual Reviews, Inc., Palo Alto, California, 1984.

[87] Won-Wook Kim. A new dynamic one-equation subgrid-scale model for large eddy simulations. Technical Report AIAA 95-0356, AIAA 33rd Aerospace Sciences Meeting and Exhibit, Reno, NV, January 1995.

[88] Won-Wook Kim and Suresh Menon. An unsteady incompressible navier-stokes solver for large eddy simulation of turbulent flows. *International Journal for Numerical Methods in Fluids*, 31:983–1017, 1999.

[89] U. Schumann. Subgrid scale model for finite difference simulation of turbulent flows in plane channels and annuli. *Journal of Computational Physics*, 18:376–404, 1975.

[90] A. Yoshizawa. Bridging between eddy-viscosity-type and second-order models using a two-scale dia. In *Ninth Symposium on Turbulent Shear Flows, Kyoto, Japan, August 16-18*, pages 23–1: 1–6, 1993.

[91] Helmut Schmidt and Ulrich Schumann. Coherent structure of the convective boundary layer derived from large-eddy simulations. *Journal of Fluid Mechanics*, 200:511–562, 1989.

[92] V. K. Chakravarthy and S. Menon. Large-eddy simulations of turbulent premixed flames in the flamelet region. *to appear Combustion Sience and Technology*, 2001.

[93] W.-W. Kim. *A New Dynamic Subgrid-Scale Model for Large-Eddy Simulation of Turbulent Flows*. PhD thesis, Georgia Institute of Technology, Atlanta, Georgia, September 1996.

[94] C. C. Nelson. *Simulations of Spatially Evolving Compressible Turbulence Using a Local Dynamic Subgrid Model*. PhD thesis, Georgia Institute of Technology, Atlanta, Georgia, December 1997.

[95] M. Germano, Ugo Piomelli, Parviz Moin, and William H. Cabot. A dynamic subgrid-scale eddy viscosity model. *Physics of Fluids A*, 3(7):1760–1765, July 1991.

[96] D. K. Lilly. A proposed modification of the germano subgrid-scale closure method. *Physics of Fluids A*, 4(3):633–635, March 1992.

[97] S. Liu, C. Meneveau, and J. Katz. On the properties of similarity subgrid scale models as deduced from measurements in a turbulent jet. *Journal of Fluid Mechanics*, 275:83, 1994.

[98] G. K. Batchelor. *The Theory of Homogeneous Turbulence*. Cambridge University Press, 1953.

[99] G. De Vahl Davis. Natural convection in a square cavity: a comparison exercise. *International Journal for Numerical Methods in Fluids*, 3:227–248, 1983.

[100] G. De Vahl Davis. Natural convection of air in a square cavity: a bench mark numerical solution. *International Journal for Numerical Methods in Fluids*, 3:249–264, 1983.

Index

Bibliography, 147
Boundary conditions, 5, 9, 35
  flux, 39
  heat flux rate, 39
  mass flux rate, 39
  natural, 37
  nodal, 35, 77
  node sets, 35
  outflow, 38, 76
  pressure, 37, 79
  segment sets, 35
  time-dependent, 99
  traction, 37, 79

Conservation equations
  energy equation, 90
  momentum equations, 85
  species transport, 90
  vector notation, 7

Element
  formulation, 99
  modified finite element formulation, 16
  reduced-integration operators, 17
Example problems, 101
  2-D channel flow, 101
  momentum-driven jet, 121
  natural convection, 111
  Poiseuille flow, 101
  sheath-flow chamber, 131
  thermal cavity, 111

Flow solver
  fluid-structure interaction, 99
  stand-alone, 99
Flow statistics, 65
  anisotropic stress tensor, 69
  derived flow statistics, 65
  fluctuating quantities, 67
  higher-order statistics, 68
  mean quantities, 67
  Reynolds averaged statistics, 66
  statistics levels, 70
    higher-order statistics (level-3), 72
    mean statistics (level-1), 71
    second moment statistics (level-2), 71
Fluid-structure interaction, 99

Governing equations, 3
  energy conservation, 5, 9
  mass conservation, 4, 8
  momentum conservation, 3, 7
  vector notation, 7
Grid parameters
  CFL number, 15, 105
  Reynolds number, 15, 105

Initial conditions, 5, 9, 95
Introduction, 1
  guide to the user's manual, 2

Keyword input, 75
  incompressible flow keywords, 75
    *BOUNDARY_OUTFLOW_CFD, 76
    *BOUNDARY_PRESCRIBED_CFD, 77
    *BOUNDARY_PRESSURE_CFD_SET, 79
    *CONTROL_CFD_AUTO, 81
    *CONTROL_CFD_GENERAL, 83
    *CONTROL_CFD_MOMENTUM, 85
    *CONTROL_CFD_PRESSURE, 88
    *CONTROL_CFD_TRANSPORT, 90
    *CONTROL_CFD_TURBULENCE, 94
    *INITIAL_CFD, 95
    *MAT_CFD_CONSTANT, 97
  shared keywords, 99
    *CONTROL_OUTPUT, 99
    *CONTROL_SOLUTION, 99
    *CONTROL_TERMINATION, 99
    *DATABASE_BINARY_D3DUMP, 99
    *DATABASE_BINARY_D3MEAN, 99
    *DATABASE_BINARY_D3PLOT, 99
    *DATABASE_BINARY_D3THDT, 99
    *DATABASE_BINARY_RUNRSF, 99
    *DATABASE_HISTORY_NODE, 99
    *DEFINE_CURVE, 99
    *SECTION_SHELL, 99
    *SECTION_SOLID, 99

LS-POST statistics levels, 73

higher-order statistics (level-3), 73
mean statistics (level-1), 73
second-moment statistics (level-2), 73

Material models
fluid properties, 97

Output
ASCII, 99
binary, 99
d3dump, 99
d3mean, 65, 99
d3plot, 99
d3thdt, 99
runrsf, 99
state database, 99
time-history database, 99

Parallelism
communication costs, 25
domain-decomposition, 22
domain-decomposition message-passing, 21
parallel assembly via message passing, 22
parallel explicit algorithm, 24
Pressure solution methods, 41
PPE solvers, 88
pressure stabilization, 42
pressure-Poisson solvers, 88
projection CG method, 44
updating Φ, 46
saddle point problems, 41
solver performance, 47
Pressure-Poisson equation
pressure levels, 40
Pressure-Poisson equation (PPE), 13, 28, 31
Projection method, 27
fully-implicit projection method, 32
first time-step, 33
general time-step, 33
start-up, 32
projection properties, 28
semi-implicit projection method, 29
Properties
fluid properties, 97

Restart
d3dump, 99
d3mean, 99
runrsf, 99

Sense switches, 139
flow solver sense switches, 139
Sets
node, 35
segment, 35
shell, 35
solid, 35
Source terms, 35
body forces, 38
buoyancy forces, 38
Start-up procedure
explicit time integration, 14
fully-implicit projection method, 32
semi-implicit projection method, 31

Time integration
explicit time integration, 13
domain-decomposition message-passing, 21
modified finite element formulation, 16
communication costs, 25
parallel explicit algorithm, 24
reduced-integration operators, 17
stability, 15
start-up, 14
projection method, 27
fully-implicit projection method, 32
projection properties, 28
semi-implicit projection method, 29
start-up, 31, 32
termination, 99
time step control, 81
time step selection, 81
Turbulence models, 57, 94
k^{sgs} model, 62
averaging, 58
dynamic k-transport model, 63
ensemble averaging, 58, 66
filtered equations, 61
filtering, 58
k-transport model, 62
LES filtering, 59
scale separation, 58
Smagorinsky subgrid-scale model, 62
space averaging, 58
spatial filtering, 59
time averaging, 58, 66
time-averaged equations, 60